4/5/1 (Item 1 from file: 351)
DIALOG(R)File 351:Derwent WPI

014185236 **Image available**
WPI Acc No: 2002-005933/ **200201**
XRPX Acc No: N02-005023
Program converter for multi-thread type microprocessor, converts conditional branch in middle of program into parallel unconditional branch using FORK command, when thread execution devices operate parallelly
Patent Assignee: NEC CORP (NIDE)
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 2001282549	A	20011012	JP 200093508	A	20000330	200201	B

Priority Applications (No Type Date): JP 200093508 A 20000330
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 2001282549	A		27	G06F-009/45	

Abstract (Basic): **JP 2001282549** A
NOVELTY - The converter converts conditional branch in the middle of a program into parallel unconditional branch using FORK command, when thread execution devices operate parallelly. A command rearrangement unit (23) rearranges the commands to be executed before and after of the FORK command such that FORK command is executed earlier.
DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:
(a) Program conversion method;
(b) Recording medium storing program for executing program conversion method
USE - For information processor such as multi-thread type microprocessor.
ADVANTAGE - Enables performing parallel execution of several threads on the middle of the program exactly, using FORK command and hence workability is enhanced.
DESCRIPTION OF DRAWING(S) - The figure shows the internal component of the program converter. (Drawing includes non-English language text).

Command rearrangement unit (23)
pp; 27 DwgNo 2/21
Title Terms: PROGRAM; CONVERTER; MULTI; THREAD; TYPE; MICROPROCESSOR; CONVERT; CONDITION; BRANCH; MIDDLE; PROGRAM; PARALLEL; UNCONDITIONAL; BRANCH; FORK; COMMAND; THREAD; EXECUTE; DEVICE; OPERATE; PARALLEL
Derwent Class: T01
International Patent Class (Main): G06F-009/45
International Patent Class (Additional): G06F-009/34; G06F-009/38; G06F-009/46
File Segment: EPI

4/5/2 (Item 1 from file: 347)
DIALOG(R)File 347:JAPIO

07054914 **Image available**
DEVICE AND METHOD FOR CONVERTING PROGRAM AND RECORDING MEDIUM

PUB. NO.: 2001-282549 A]
PUBLISHED: October 12, 2001 (20011012)
INVENTOR(s): SAKAI JUNJI
APPLICANT(s): NEC CORP

APPL. NO.: 2000-093508 [JP 200093508]
FILED: March 30, 2000 (20000330)
INTL CLASS: G06F-009/45; G06F-009/34; G06F-009/38; G06F-009/46

ABSTRACT

PROBLEM TO BE SOLVED: To provide a program conversion device performing paralleling at an intermediate program level for a multithread microprocessor.

SOLUTION: A paralleling device 11 is provided with a FORK part deciding part 21, a register allocation part 22 and an instruction rearranging part 23. The FORK part deciding part 21 decides a FORK part and a FORK system based on a result obtained by trying register allocation in the register allocation part 22, the number of memory data depending parts and branch probability and data dependency generation frequency, which are obtained from a profile information file 5, with respect to the inputted intermediate program. The instruction rearranging part 23 rearranges instructions before and after a FORK instruction following the decision.

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号
特開2001－282549
（P2001－282549A）
(43)公開日　平成13年10月12日(2001.10.12)

(51)Int.Cl.[7]		識別記号	FI			テーマコード(参考)
Ｇ０６Ｆ	9/45		Ｇ０６Ｆ	9/34	３３０	５Ｂ０１３
	9/34	３３０		9/38	３３０Ｋ	５Ｂ０３３
	9/38	３３０		9/46	３６０Ｂ	５Ｂ０８１
	9/46	３６０		9/44	３２２Ｆ	５Ｂ０９８

審査請求　有　　請求項の数33　ＯＬ　（全 27 頁）

(21)出願番号　特願2000－93508(P2000－93508)

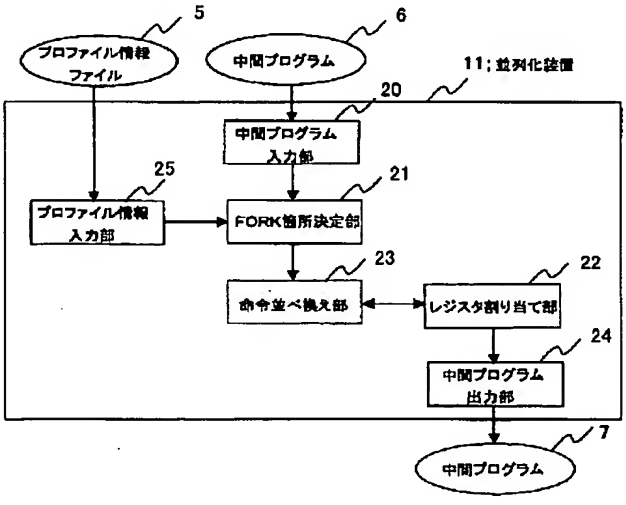
(22)出願日　平成12年３月30日(2000.3.30)

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(72)発明者　酒井　淳嗣
東京都港区芝五丁目７番１号　日本電気株式会社内
(74)代理人　100080816
弁理士　加藤　朝道

Ｆターム(参考)　5B013 BB18
5B033 AA14 BE00
5B081 CC23 CC25 CC32
5B098 AA10 GA05 GC14

(54)【発明の名称】　**プログラム変換装置及び方法並びに記録媒体**

(57)【要約】
【課題】中間プログラムレベルでマルチスレッドマイクロプロセッサ向けの並列化を行うプログラム変換装置の提供。
【解決手段】並列化装置１１は、ＦＯＲＫ箇所決定部２１とレジスタ割り当て部２２と命令並べ換え部２３を備え、入力した中間プログラムに対し、ＦＯＲＫ箇所決定部２１はレジスタ割り当て部２２でレジスタ割り当てを試行した結果とメモリデータ依存箇所の数とプロファイル情報ファイル５から得た分岐確率及びデータ依存発生頻度をもとに、ＦＯＲＫ箇所と、ＦＯＲＫ方式を決定し、命令並べ換え部２３は、その決定に沿ってＦＯＲＫ命令前後の命令を並べ換える。



1

【特許請求の範囲】

【請求項１】複数のプログラムカウンタと、複数のスレッド実行装置と、を備え、前記複数のスレッド実行装置は、前記複数のプログラムカウンタに従って、複数のスレッドの命令を、同時に、フェッチ、解釈、実行し、

スレッド生成時以降に、レジスタセットに及ぼした変更を、後に取り消し可能な制御投機的モードでスレッドを実行すること、及び、

自スレッドがメモリ領域から値をロードした後に、自スレッドを生成した親スレッドが同一メモリ領域に値をストアした場合に、自スレッドの少なくとも当該ロード以降の処理結果を破棄し、それらの処理を再実行するデータ依存投機的モードでスレッドを実行すること、が可能とされ、

命令セットとして、

前記スレッド実行装置で実行中のスレッドが制御投機的モードの新たなスレッドを生成すること、

指定された条件が成立していれば自スレッドを終了させると共に、自スレッドが生成した制御投機的モードのスレッドの制御投機的モードを解除すること、

生成した制御投機的モードのスレッドを破棄すること、

自スレッドが生成するスレッドが指定されたアドレスのメモリ領域からのロードを行う際に、その動作を一時停止させることをあらかじめ指示すること、

指定されたメモリアドレスに対する前記ロード一時停止指示を解除すること、前記スレッド実行装置で実行中のスレッドがデータ依存投機的モードの新たなスレッドを生成すること、及び、

自スレッドが生成したデータ依存投機的モードのスレッドのデータ依存投機的モードを解除すること、

が、単一又は高々数個の機械語命令の組合せで実行できる命令セットを有するマルチスレッドプロセッサに対して、与えられた原始プログラムを前記マルチスレッドプロセッサ向けに変換するプログラム変換装置であって、

並列化に先立ってレジスタ割り当てを試み、中間プログラム上の各変数、及び中間項のレジスタ割り当て状況を予測するレジスタ割り当て試行部と、

前記レジスタ割り当て試行部でのレジスタ割り当て試行結果に基づいて、前記中間プログラムにおける条件分岐部分をスレッド生成命令を用いた並列コードに変換するか否か決定し、及び、並列コードでの並列実行方式の決定を行うフォーク（ＦＯＲＫ）箇所決定部と、

前記フォーク（ＦＯＲＫ）箇所決定部での決定結果に基づいて、前記中間プログラム中の条件分岐部分を、スレッド生成命令を用いた並列コードに変換し、前記レジスタ割り当て試行結果を参照して、スレッド間のメモリを介したデータ依存関係を保証する命令を、前記スレッド生成命令の前後に挿入すると共に、スレッド生成が早い段階で行われるように、前記スレッド生成命令の前後の命令を並べ換える命令並べ換え部と、

2

並列化され並べ換えられた命令列に対して、物理レジスタが割り当てられるか否かに関して、前記レジスタ割り当ての試行時と同じ割り当て結果となるように確定的なレジスタ割り当てを行うレジスタ割り当て部と、

を備える、ことを特徴とするプログラム変換装置。

【請求項２】前記フォーク（ＦＯＲＫ）箇所決定部が、現在処理対象としている中間プログラム中の基本ブロックから、該基本ブロック末尾にある条件分岐命令の分岐先基本ブロック各々へのメモリを介したデータ依存関係を調査し、

分岐先の各々について、分岐先基本ブロック中にあってデータ依存を引き起こしているメモリ参照命令のうち、最も先頭にある命令の、該分岐先基本ブロック先頭からの命令ステップ数を数え、

前記命令ステップ数が大きな側の分岐先基本ブロックを、並列実行させる新たなスレッドとして選択する、ことを特徴とする請求項１に記載のプログラム変換装置。

【請求項３】前記フォーク（ＦＯＲＫ）箇所決定部が、分岐先基本ブロックにてメモリを介したデータ依存命令の位置を求める際、前記命令ステップ数に代えて、各命令の推定実行サイクル数を積算した値を用いる、ことを特徴とする請求項２に記載のプログラム変換装置。

【請求項４】請求項１、請求項２または請求項３に記載のプログラム変換装置において、

初回に原始プログラムから目的プログラムに変換する際に、前記プログラム変換装置内の中間プログラムでの基本ブロックと、出力する目的プログラムでの機械命令アドレスとの対応をとるためのアドレス対応情報を、前記目的プログラムと併せて出力し、

目的プログラム実行装置が、前記目的プログラムと前記アドレス対応情報を読み込んで、前記目的プログラムを実行すると共に、前記目的プログラムの実行時の基本ブロック間の分岐プロファイル情報と、基本ブロック間でメモリを介して発生したデータ依存情報とを含むプロファイル情報を出力し、

次に、前記プログラム変換装置が、原始プログラムを並列化して目的プログラムに変換する際に、前記フォーク（ＦＯＲＫ）箇所決定部が、前記プロファイル情報を参照して、条件分岐において、制御の流れる確率が高い分岐先基本ブロック及び条件分岐において、データ依存が発生する確率が低い分岐先基本ブロックを並列実行する新スレッドとして優先的に選択する、ことを特徴とする、プログラム変換装置。

【請求項５】前記フォーク（ＦＯＲＫ）箇所決定部が、並列実行する新スレッドの実行開始点として選択した条件分岐先基本ブロックに対し、中間プログラムでのメモリを介したデータ依存を解析した結果と、前記プロファイル情報から得たデータ依存発生確率とに基づき、データ依存を引き起こす異なるメモリアドレスの箇所の数が定められた数より少ない場合には、新スレッドが該メモ

リアドレスからロードする動作を一時停止させるように命令を生成し、

一方、データ依存を引き起こす異なるメモリアドレスの箇所の数が定められた数より多い場合には、データ依存発生確率が定められた確率より低いか否かを調べ、確率が低い場合には、データ依存投機的モードで新スレッドを生成するように命令を生成し、

確率が高い場合には、当該箇所での並列化変換を取りやめるように制御する、ことを特徴とする請求項４に記載のプログラム変換装置。

【請求項６】前記フォーク（ＦＯＲＫ）箇所決定部が、現在処理対象としている中間プログラム中の基本ブロックから、該基本ブロック末尾にある条件分岐命令の分岐先基本ブロック各々へのメモリを介したデータ依存関係を調査し、該調査したデータ依存関係と、前記プロファイル情報から得た条件分岐確率とを総合した結果、当該条件分岐での分岐先基本ブロック各々に関して、分岐確率の間に定められた以上の差異がなく、且つ、メモリを介したデータ依存発生時期の早遅にも定められた以上の差異がない場合に、当該条件分岐部分を並列化しないように決定する、ことを特徴とする請求項４または請求項５に記載のプログラム変換装置。

【請求項７】原始（ソース）プログラムを構文解析して中間プログラムを生成する構文解析部と、

前記中間プログラムに対して並列化を含む最適化処理を行う並列化部と、

前記並列化装置で最適化済みの中間プログラムからターゲットプロセッサ装置向きの命令コードからなる目的プログラム（オブジェクトコード）を生成出力するコード生成部と、

を備えたプログラム変換装置において、

前記並列化部が、

前記中間プログラムを読み込んで制御フローやデータフローを解析する中間プログラム入力部と、

並列化に先立ってレジスタ割り当てを試み、中間プログラム上の各変数、及び中間項のレジスタ割り当て状況を予測するとともに、レジスタの割り当てを実行するレジスタ割り当て部と、

前記レジスタ割り当て試行結果に基づいて、前記中間プログラムにおける条件分岐部分をスレッド生成命令を用いた並列コードに変換する箇所を決定するフォーク（ＦＯＲＫ）箇所決定部と、

前記ＦＯＲＫ箇所決定部で決定された並列化箇所とデータフローなどの情報から、前記並列化箇所前後の命令の並べ換えを行う命令並べ換え部と、

前記並列化を含む変換を終えた命令列を再度中間プログラム形式で出力する中間プログラム出力部と、

を備えた、ことを特徴とするプログラム変換装置。

【請求項８】前記並列化部が、前記ターゲットプロセッサ装置で前記目的プログラムを実行して出力されるプロファイル情報を入力して内部形式に変換するプロファイル情報入力部を備え、

前記フォーク（ＦＯＲＫ）箇所決定部が、前記レジスタ割り当て試行結果と、前記プロファイル情報に基づき、前記中間プログラムにおける条件分岐部分をスレッド生成命令を用いた並列コードに変換する箇所を決定するとともに、並列コードによる並列実行方式を決定する、ことを特徴とする請求項７記載のプログラム変換装置。

【請求項９】前記命令並べ換え部が、前記フォーク（ＦＯＲＫ）箇所決定部での決定結果に基づいて、前記中間プログラム中の条件分岐部分を、スレッド生成命令を用いた並列コードに変換し、前記レジスタ割り当て試行結果を参照して、スレッド間のメモリを介したデータ依存関係を保証する命令を、前記スレッド生成命令の前後に挿入すると共に、スレッド生成が早い段階で行われるように、前記スレッド生成命令の前後の命令を並べ換える、ことを特徴とする請求項７記載のプログラム変換装置。

【請求項１０】前記レジスタ割り当て部が、並列化され並べ換えられた命令列に対して、物理レジスタが割り当てられるか否かに関して前記レジスタ割り当ての試行時と同じ割り当て結果となるように確定的なレジスタ割り当てを行う、ことを特徴とする請求項７記載のプログラム変換装置。

【請求項１１】原始プログラムを構文解析部で構文解析して得られる中間プログラムに対して、ターゲットプロセッサ装置向けに、並列化を含む最適化処理を行うプログラム変換装置において、

前記中間プログラム上で、前記ターゲットプロセッサ装置でのレジスタの割り当てを試行し、レジスタ割り当て情報を実際の割り当てに先行して取得するレジスタ割り当て試行手段と、

前記中間プログラムに対して、前記ターゲットプロセッサ装置でのメモリを介して発生するデータ依存の距離の計算を行う手段と、

前記中間プログラム上で、前記メモリを介するデータ依存の距離を考慮して、フォーク（ＦＯＲＫ）先を決定し、条件分岐をスレッド生成命令に置き換える手段と、

前記レジスタ割り当て試行結果を参照して、前記中間プログラム上で、前記スレッド生成命令前後の命令の並べ換えを行う手段と、

を備えたことを特徴とするプログラム変換装置。

【請求項１２】前記条件分岐をスレッド生成命令に置き換える手段が、

前記条件分岐の２つの分岐先の各々に対して、各中間項及び変数のデータ依存の距離の最小値を計算する手段と、

前記条件分岐の２つの分岐に関して、それぞれ求めた２つのデータ依存の距離の最小値を比較し、両者に所定値以上の差がある場合、前記データ依存の距離の最小値の

大きい側の分岐方向をフォーク（ＦＯＲＫ）先とし、該条件分岐箇所をフォーク（ＦＯＲＫ）箇所に選定し、一方、前記データ依存の距離の最小値に前記所定値以上の差がない場合には、元の中間プログラム内で分岐先であった側をフォーク（ＦＯＲＫ）先とし、該条件分岐箇所をフォーク（ＦＯＲＫ）箇所候補に選定する手段と、を備えた、ことを特徴とする請求項１１記載のプログラム変換装置。

【請求項１３】請求項１１記載のプログラム変換装置において、

前記プログラム変換装置から出力される目的プログラムを実行するプロセッサ装置から出力されるプロファイル情報を入力し、前記プロファイル情報から、条件分岐確率及びデータ依存発生頻度を計算する手段と、

前記データ依存の距離と、前記条件分岐確率及び前記データ依存発生頻度と、データ依存を引き起こす互いに異なるメモリアドレスの箇所の数から、フォーク（ＦＯＲＫ）先及びデータ依存保証方式を決定し、条件分岐をスレッド生成命令に置き換える手段と、を備えたことを特徴とするプログラム変換装置。

【請求項１４】前記ターゲットプロセッサ装置が、複数のプログラムカウンタと、複数のスレッド実行装置と、を備え、前記複数のスレッド実行装置は、前記複数のプログラムカウンタに従って、複数のスレッドの命令を、同時に、フェッチ、解釈、実行し、

スレッド生成時以降に、レジスタセットに及ぼした変更を、後に取り消し可能な制御投機的モードでスレッドを実行すること、及び、

自スレッドがメモリ領域から値をロードした後に、自スレッドを生成した親スレッドが同一メモリ領域に値をストアした場合に、自スレッドの少なくとも当該ロード以降の処理結果を破棄し、それらの処理を再実行するデータ依存投機的モードでスレッドを実行すること、が可能とされ、

命令セットとして、

前記スレッド実行装置で実行中のスレッドが制御投機的モードの新たなスレッドを生成すること、

指定された条件が成立していれば自スレッドを終了させると共に、自スレッドが生成した制御投機的モードのスレッドの制御投機的モードを解除すること、

生成した制御投機的モードのスレッドを破棄すること、

自スレッドが生成するスレッドが指定されたアドレスのメモリ領域からのロードを行う際に、その動作を一時停止させることをあらかじめ指示すること、

指定されたメモリアドレスに対する前記ロード一時停止指示を解除すること、

前記スレッド実行装置で実行中のスレッドがデータ依存投機的モードの新たなスレッドを生成すること、及び、

自スレッドが生成したデータ依存投機的モードのスレッドのデータ依存投機的モードを解除すること、

が、単一又は高々数個の機械語命令の組合せで実行できる命令セットを有するマルチスレッドプロセッサよりなる、ことを特徴とする請求項７乃至１３のいずれか一に記載のプログラム変換装置。

【請求項１５】原始プログラムをコンパイルしてマルチスレッド型のターゲットプロセッサ装置向けの目的プログラムを出力するプログラム変換装置において、構文解析の結果出力される中間プログラムに対して並列化を含む最適化処理を行う方法であって、

（ａ）並列化に先立ってレジスタ割り当てを試み、中間プログラム上の各変数、及び中間項のレジスタ割り当て状況を予測するレジスタ割り当て試行ステップと、

（ｂ）前記レジスタ割り当て試行結果に基づいて、前記中間プログラムにおける条件分岐部分をスレッド生成命令を用いた並列コードに変換するか否かの決定を行うか、もしくは、並列コードに変換するか否かの決定と並列コードに変換する場合その並列実行方式の決定を行うフォーク（ＦＯＲＫ）箇所決定ステップと、

（ｃ）前記フォーク（ＦＯＲＫ）箇所決定ステップでの決定結果に基づいて、前記中間プログラム中の条件分岐部分を、スレッド生成命令を用いた並列コードに変換し、前記レジスタ割り当て試行結果を参照して、スレッド間のメモリを介したデータ依存関係を保証する命令を、前記スレッド生成命令の前後に挿入すると共に、スレッド生成が早い段階で行われるように、前記スレッド生成命令の前後の命令を並べ換える命令並べ換えステップと、

（ｄ）並列化され並べ換えられた命令列に対して前記レジスタ割り当ての試行時と同じ割り当て結果となるように確定的なレジスタ割り当てを行うレジスタ割り当てステップと、

を含む、ことを特徴とするプログラム並列化方法。

【請求項１６】前記フォーク（ＦＯＲＫ）箇所決定ステップが、前記条件分岐の２つの分岐先の各々に対して、各中間項及び変数のデータ依存の距離の最小値を計算し、前記条件分岐の２つの分岐に関して、それぞれ求めた２つのデータ依存の距離の最小値を比較し、両者に所定値以上の差がある場合、前記データ依存の距離の最小値の大きい側の分岐方向をフォーク（ＦＯＲＫ）先とし、該条件分岐箇所をフォーク（ＦＯＲＫ）箇所に選定し、一方、前記データ依存の距離の最小値に前記所定値以上の差がない場合には、元の中間プログラム内で分岐先であった側をフォーク（ＦＯＲＫ）先とし、該条件分岐箇所をフォーク（ＦＯＲＫ）箇所候補に選定する、ことを特徴とする請求項１５記載のプログラム並列化方法。

【請求項１７】前記フォーク（ＦＯＲＫ）箇所決定ステップが、現在処理対象としている中間プログラム中の、分岐や合流のない一続きのブロック（「基本ブロック」という）から、該基本ブロック末尾にある条件分岐命令

の分岐先基本ブロック各々へのメモリを介したデータ依存関係を調査し、

分岐先の各々について、分岐先基本ブロック中にあってデータ依存を引き起こしているメモリ参照命令のうち、最も先頭にある命令の、該分岐先基本ブロック先頭からの命令ステップ数を数え、

前記命令ステップ数が大きな側の分岐先基本ブロックを、並列実行させる新たなスレッドとして選択する、ことを特徴とする請求項１５に記載のプログラム並列化方法。

【請求項１８】前記フォーク（ＦＯＲＫ）箇所決定ステップが、分岐先基本ブロックにてメモリを介したデータ依存命令の位置を求める際、前記命令ステップ数に代えて、各命令の推定実行サイクル数を積算した値を用いる、ことを特徴とする請求項１７に記載のプログラム並列化方法。

【請求項１９】前記プログラム変換装置で、まず、原始プログラムから目的プログラムに変換する際に、中間プログラムでの基本ブロックと、出力する目的プログラムでの機械命令アドレスとの対応をとるためのアドレス対応情報を、前記目的プログラムと併せて出力し、

前記目的プログラムを実行するプロセッサ装置が、前記目的プログラムと前記アドレス対応情報を読み込んで、前記目的プログラムを実行すると共に、前記目的プログラムの実行時の基本ブロック間の分岐プロファイル情報と、基本ブロック間でメモリを介して発生したデータ依存情報とを含むプロファイル情報を出力し、

次に、前記プログラム変換装置が、原始プログラムを並列化して、目的プログラムに変換する際に、前記フォーク（ＦＯＲＫ）箇所決定ステップにおいて、前記プロファイル情報を参照して、条件分岐において、制御の流れる確率が高い分岐先基本ブロック及び条件分岐において、データ依存が発生する確率が低い分岐先基本ブロックを並列実行する新スレッドとして優先的に選択する、ことを特徴とする、請求項１５に記載のプログラム並列化方法。

【請求項２０】前記フォーク（ＦＯＲＫ）箇所決定ステップが、並列実行する新スレッドの実行開始点として選択した条件分岐先基本ブロックに対し、中間プログラムでのメモリを介したデータ依存を解析した結果と、前記プロファイル情報から得たデータ依存発生確率とに基づき、データ依存を引き起こす異なるメモリアドレスの箇所の数が定められた数よりも少ない場合には、新スレッドが該メモリアドレスからロードする動作を一時停止させるように命令を生成し、

一方、データ依存を引き起こす異なるメモリアドレスの箇所の数が定められた数より多い場合には、データ依存発生確率が定められた確率より低いか否かを調べ、確率が低い場合には、データ依存投機的モードで新スレッドを生成するように命令を生成し、

確率が高い場合には、当該箇所での並列化変換を取りやめるように制御する、ことを特徴とする、請求項１９に記載のプログラム並列化方法。

【請求項２１】前記フォーク（ＦＯＲＫ）箇所決定ステップが、現在処理対象としている中間プログラム中の基本ブロックから、該基本ブロック末尾にある条件分岐命令の分岐先基本ブロック各々へのメモリを介したデータ依存関係を調査し、該調査したデータ依存関係と、前記プロファイル情報から得た条件分岐確率とを総合した結果、当該条件分岐での分岐先基本ブロック各々に関して、分岐確率の間に、予め定められた以上の差異がなく、且つ、メモリを介したデータ依存発生時期の早遅にも、予め定められた以上の差異がない場合に、当該条件分岐部分を並列化しないように決定する、ことを特徴とする請求項１９に記載のプログラム並列化方法。

【請求項２２】前記フォーク（ＦＯＲＫ）箇所決定ステップが、

（１）条件分岐命令が前記中間プログラム中のループ構造の戻り枝に相当する分岐命令であるか否かを判定するステップと、

（２）前記条件分岐命令がループ戻り枝に相当する分岐命令である場合、当該戻り枝方向、即ちループ継続方向をＦＯＲＫ先として、この条件分岐箇所を、ＦＯＲＫ箇所に選定するステップと、

（３）前記条件分岐命令が、ループ戻り枝分岐ではない場合、条件分岐の２つの分岐先の各々に対して、各中間項／変数のデータ依存の距離の最小値を計算するステップと、

（４）前記条件分岐の２つの分岐に関してそれぞれ求めた２つのデータ依存の距離の最小値を比較し、両者に所定値以上の差があるか否かを判定するステップと、

（５）前記２つのデータ依存の距離の最小値に前記所定値以上の差がある場合、データ依存の距離の最小値の大きい側の分岐方向をＦＯＲＫ先とし、この条件分岐箇所を、ＦＯＲＫ箇所に選定するステップと、

（６）データ依存の距離の最小値に前記所定値以上の差がない場合、元の中間プログラム内で分岐先であった側（分岐命令のｔａｋｅｎ側）をＦＯＲＫ先とし、この条件分岐箇所をＦＯＲＫ候補に選定するステップと、

を含むことを特徴とする請求項１５に記載のプログラム並列化方法。

【請求項２３】前記データ依存の距離が、現在、処理対象としている基本ブロック内で定義され、その分岐先で参照される可能性がある中間項及び変数のうち、メモリ上に配置されると見込まれる中間項、及び変数の各々に対して、分岐先の基本ブロックの中で、該メモリ参照命令が、先頭からどれぐらいの位置にあるかを、中間プログラム中のステップ数で表したものである、ことを特徴とする請求項２２に記載のプログラム並列化方法。

【請求項２４】前記データ依存の距離を求める際、各命

令が目的アーキテクチャのプロセッサ上で実行される際に要すると推定されるサイクル数を用いる、ことを特徴とする請求項２２に記載のプログラム並列化方法。

【請求項２５】前記命令並べ換えステップが、

（１）前記中間プログラム中の各中間項及び変数がレジスタに対応付けられるかメモリに対応付けられるかレジスタの割り当て状況を調べるステップと、

（２）現在処理対象としている基本ブロックの末尾にある分岐命令を、制御投機モードＦＯＲＫ命令に置換し、その際、制御投機モードＦＯＲＫ命令のオペランドであるＦＯＲＫ先は、前記ＦＯＲＫ箇所決定ステップで選択されたＦＯＲＫ先とするステップと、

（３）前記中間プログラム中で、制御投機モードＦＯＲＫ命令の直前にある分岐条件式を、制御投機モードＦＯＲＫ命令直後に移動させると共に、該移動先の直後すなわち当該基本ブロックの末尾に、分岐条件成立時には自スレッドを終了させて子スレッドを確定モードすなわち非制御投機モードに移行させ、分岐条件非成立時には子スレッドを破棄させて自スレッドが後続命令列の実行を続ける、一群の命令列を、挿入するステップと、

（４）現在処理対象としている基本ブロックにおいて、前記制御投機モードＦＯＲＫ命令よりも手前、即ち、上流側にある命令文のうち、メモリに対応付けられた中間項及び変数への代入となる文の各々について、その代入文を、前記制御投機モードＦＯＲＫ命令よりも後方、即ち、下流側へ移動させると共に、前記制御投機モードＦＯＲＫ命令の直前に、ブロック設定命令を挿入し、代入文の移動先の直後に、ブロック解除命令を挿入するステップと、

（５）前記ステップ（２）のＦＯＲＫ変換処理で仮定したレジスタ割り当て状況に、レジスタを割り当てるように指示するステップと、

を含む、ことを特徴とする請求項１５に記載のプログラム並列化方法。

【請求項２６】前記フォーク（ＦＯＲＫ）箇所決定ステップが、

（１）条件分岐命令が前記中間プログラム中のループ構造の戻り枝に相当する分岐命令であるか否かを判定するステップと、

（２）ループ構造の戻り枝に相当する場合、前記戻り枝方向をＦＯＲＫ先として仮決定するステップと、

（３）入力したプロファイル情報に基づいて、当該条件分岐命令のｔａｋｅｎ（分岐成立）側と、ｆａｌｌｔｈｒｏｕｇｈ（フォールスルー）側が選択される確率と、を計算するステップと、

（４）前記計算された２つ分岐の確率間に所定の基準値以上の違いがあるかどうかを判定するステップと、

（５）前記２つ分岐の確率の違いが前記基準値を越えていれば、確率の高い側を、ＦＯＲＫ先として仮決定するステップと、

（６）条件分岐の２つの分岐先各々に対して、データ依存の距離の最小値を計算するステップと、

（７）条件分岐の２つの分岐に関して求めた２つのデータ依存の距離の最小値を比較し、両者に所定値以上の差があるか否かを判定するステップと、

（８）２つのデータ依存の距離の最小値に前記所定値以上の差があるか又はデータ依存がない場合、データ依存の距離の最小値の大きい側の分岐方向をＦＯＲＫ先として決定するステップと、

（９）前記ステップ（２）又は（５）で仮決定されたＦＯＲＫ先に対して、データ依存の距離の最小値を計算し、仮決定したＦＯＲＫ先側のデータ依存の距離の最小値が所定値以上あるか否かを判定するステップと、

（１０）仮決定したＦＯＲＫ先側のデータ依存の距離の最小値に所定値以上の差があるか、メモリを介したデータ依存がなければ、ステップ（２）又はステップ（５）で仮決定したＦＯＲＫ先を、正式なＦＯＲＫ先として確定するステップと、

（１１）前記データの依存の距離の最小値が一定水準に満たないと判断された場合には、当該基本ブロックを、ＦＯＲＫ箇所から除外するステップと、

（１２）入力したプロファイル情報からデータ依存発生頻度を計算するステップと、

（１３）データの依存発生頻度が一定水準より高いか否かを判断し、低い場合、前記中間プログラム中で、ＦＯＲＫ元基本ブロックから、ＦＯＲＫ先基本ブロックへのデータ依存を引き起こし得るメモリ上の中間項／変数の個数を数え挙げ、それが一定水準より多いか否かを判定し、データ依存箇所数が一定水準より多ければＤＳＰ方式によるＦＯＲＫを用い、そうでなければＢＬＯＣＫ方式によるＦＯＲＫを用いることとし、その情報を、中間プログラム中のＦＯＲＫ命令に付与するステップと、

（１４）データの依存発生頻度が一定水準より高い場合、データ依存しているメモリ上の変数の個数を数え挙げ、その数が、一定水準より少なければ、ＢＬＯＣＫ方式によるＦＯＲＫを用い、一定水準より多ければ、当該基本ブロックはＦＯＲＫ候補から外すステップと、

を含む、ことを特徴とする請求項１５に記載のプログラム並列化方法。

【請求項２７】前記命令並べ換えステップが、

（１）前記中間プログラム中の各中間項及び変数がレジスタに対応付けられるかメモリに対応付けられるかレジスタの割り当て状況を調べるステップと、

（２）現在処理対象としている基本ブロックの末尾にある分岐命令を、制御投機モードＦＯＲＫ命令に置換し、その際、制御投機ＦＯＲＫ命令のオペランドであるＦＯＲＫ先は、前記ＦＯＲＫ箇所決定ステップで選択されたＦＯＲＫ先とするステップと、

（３）前記中間プログラム中で、制御投機ＦＯＲＫ命令の直前にある分岐条件式を、制御投機ＦＯＲＫ命令直後

に移動させると共に、該移動先の直後、即ち当該基本ブロックの末尾に分岐条件成立時には自スレッドを終了させて子スレッドを確定モードすなわち非制御投機モードに移行させ、分岐条件非成立時には、子スレッドを破棄させて自スレッドが後続命令列の実行を続ける一群の命令列を挿入するステップと、

（４）前記ＦＯＲＫ箇所決定ステップが決定した当該ＦＯＲＫ箇所のＦＯＲＫ時データ保証方式がＢＬＯＣＫ方式かＤＳＰ方式かをチェックするステップと、

（５）ＢＬＯＣＫ方式の場合、ＦＯＲＫ前のメモリストア文をＦＯＲＫ後に移動すると共に、必要なブロック設定及びブロック解除命令を挿入し、移動の際には、データ依存関係を検査し、命令実行順序が入れ替わっても演算結果が不変となるもののみ移動させるステップと、

（６）ＤＳＰ方式の場合、メモリに対応付けられた中間項への代入文を、ＦＯＲＫ命令後に移動させ、データ依存投機モードでＦＯＲＫを行うように、前記ステップ（２）で置換作成されたＦＯＲＫ命令を修正するステップと、

（７）前記ステップ（２）のＦＯＲＫ変換処理で仮定したレジスタ割り当て状況をレジスタ割り当てるように指示するステップと、

を含む、ことを特徴とする請求項１５に記載のプログラム並列化方法。

【請求項２８】原始プログラムをコンパイルしてマルチスレッドプロセッサ装置向けの目的プログラムを生成出力するコンパイラにおいて、構文解析の結果出力される中間プログラムに対して並列化を含む最適化処理であって、

（ａ）並列化に先立ってレジスタ割り当てを試み、中間プログラム上の各変数、及び中間項のレジスタ割り当て状況を予測するレジスタ割り当て試行処理と、

（ｂ）前記レジスタ割り当て試行結果に基づいて、前記中間プログラムにおける条件分岐部分をスレッド生成命令を用いた並列コードに変換するか否かの決定を行うか、もしくは、並列コードに変換するか否かの決定と並列コードに変換する場合その並列実行方式の決定を行うフォーク（ＦＯＲＫ）箇所決定処理と、

（ｃ）前記フォーク（ＦＯＲＫ）箇所決定部での決定結果に基づいて、前記中間プログラム中の条件分岐部分を、スレッド生成命令を用いた並列コードに変換し、前記レジスタ割り当て試行結果を参照して、スレッド間のメモリを介したデータ依存関係を保証する命令を、前記スレッド生成命令の前後に挿入すると共に、スレッド生成が早い段階で行われるように、前記スレッド生成命令の前後の命令を並べ換える命令並べ換え処理と、

（ｄ）並列化され並べ換えられた命令列に対して、物理レジスタが割り当てられるか否かに関して、前記レジスタ割り当ての試行時と同じ割り当て結果となるように確定的なレジスタ割り当てを行うレジスタ割り当て処理と、

の前記（ａ）乃至（ｄ）の最適化処理をコンピュータで実行するためのプログラムを記録した記録媒体。

【請求項２９】請求項２８記載の記録媒体において、

前記フォーク（ＦＯＲＫ）箇所決定処理が、現在処理対象としている中間プログラム中の基本ブロックから、該基本ブロック末尾にある条件分岐命令の分岐先基本ブロック各々へのメモリを介したデータ依存関係を調査し、分岐先の各々について、分岐先基本ブロック中にあってデータ依存を引き起こしているメモリ参照命令のうち、最も先頭にある命令の、該分岐先基本ブロック先頭からの命令ステップ数を数え、

前記命令ステップ数が大きな側の分岐先基本ブロックを、並列実行させる新たなスレッドとして選択する、処理を含み、前記処理を、前記コンピュータで実行するためのプログラムを記録した記録媒体。

【請求項３０】請求項２８記載の記録媒体において、

前記フォーク（ＦＯＲＫ）箇所決定処理において、分岐先基本ブロックにてメモリを介したデータ依存命令の位置を求める際、前記命令ステップ数に代えて、各命令の推定実行サイクル数を積算した値を用い処理を、前記コンピュータで実行するためのプログラムを記録した記録媒体。

【請求項３１】請求項２８記載の記録媒体において、

（ｆ）前記コンパイラで、まず、原始プログラムから目的プログラムに変換する際に、中間プログラムでの基本ブロックと、出力する目的プログラムでの機械命令アドレスとの対応をとるためのアドレス対応情報を、前記目的プログラムと併せて出力する処理と、

前記目的プログラムを実行するプロセッサ装置が、前記目的プログラムと前記アドレス対応情報を読み込んで、前記目的プログラムを実行すると共に、前記目的プログラムの実行時の基本ブロック間の分岐プロファイル情報と、基本ブロック間でメモリを介して発生したデータ依存情報とを含むプロファイル情報を出力し、

前記コンパイラが、原始プログラムを並列化して、目的プログラムに変換する際に、

（ｇ）前記フォーク（ＦＯＲＫ）箇所決定処理において、前記プロファイル情報を参照して、条件分岐において、制御の流れる確率が高い分岐先基本ブロック及び条件分岐において、データ依存が発生する確率が低い分岐先基本ブロックを並列実行する新スレッドとして優先的に選択する処理、

の前記（ｆ）及び（ｇ）の処理を前記コンピュータで実行するためのプログラムを記録した記録媒体。

【請求項３２】請求項３１記載の記録媒体において、

前記フォーク（ＦＯＲＫ）箇所決定処理が、並列実行する新スレッドの実行開始点として選択した条件分岐先基本ブロックに対し、中間プログラムでのメモリを介したデータ依存を解析した結果と、前記プロファイル情報か

ら得たデータ依存発生確率とに基づき、データ依存を引き起こす異なるメモリアドレスの箇所の数が定められた数より少ない場合には、新スレッドが該メモリアドレスからロードする動作を一時停止させるように命令を生成し、

一方、データ依存を引き起こす異なるメモリアドレスの箇所の数が定められた数より多い場合には、データ依存発生確率が定められた確率より低いか否かを調べ、確率が低い場合には、データ依存投機的モードで新スレッドを生成するように命令を生成し、

確率が高い場合には、当該箇所での並列化変換を取りやめるように制御する、

処理を含み、前記処理を前記コンピュータで実行するためのプログラムを記録した記録媒体。

【請求項３３】請求項３１記載の記録媒体において、

前記フォーク（ＦＯＲＫ）箇所決定処理が、現在処理対象としている中間プログラム中の基本ブロックから、該基本ブロック末尾にある条件分岐命令の分岐先基本ブロック各々へのメモリを介したデータ依存関係を調査し、該調査したデータ依存関係と、前記プロファイル情報から得た条件分岐確率とを総合した結果、当該条件分岐での分岐先基本ブロック各々に関して、分岐確率の間に定められた以上の差異がなく、且つ、メモリを介したデータ依存発生時期の早遅にも定められた以上の差異がない場合に、当該条件分岐部分を並列化しないように決定する、処理を含み、前記処理を前記コンピュータで実行するためのプログラムを記録した記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数の機械命令を同時に実行できるマルチスレッド型マイクロプロセッサのためのプログラム変換技術に関し、特に、小さな粒度でも実行効率の高い並列プログラムを生成するためのプログラム変換技術に関する。

【０００２】

【従来の技術】情報処理装置において、プログラムの実行性能を高めるための一技法として、従来より、プログラムを複数の命令流即ちスレッドに分割し、それらのスレッドを並列実行するマルチスレッド実行技術が採用されている。

【０００３】それぞれに実行ユニットを有する複数のプロセッサエレメント（プロセッサ要素）を密に結合したマルチスレッドマイクロプロセッサは、スレッドの生成、同期等の処理を低コストで行えるという特徴を備えており、逐次性が高くスレッド粒度（スレッドのサイズ）が小さいプログラムに対しても、十分な並列処理効果を出すことができる。

【０００４】このようなマルチスレッドマイクロプロセッサのアーキテクチャとして、

・文献１（「Ｏｎ－Ｃｈｉｐ　Ｍｉｃｒｏｐｒｏｃｅｓｓｏｒ指向制御並列アーキテクチャＭＵＳＣＡＴの提案」（鳥居他、並列処理シンポジウムＪＳＰＰ９７論文集、情報処理学会、第２２９－２３６頁、Ｍａｙ１９９７））、

・文献２（「オンチップ制御並列プロセッサＭＵＳＣＡＴの提案」（鳥居他、情報処理学会論文誌、Ｖｏｌ．３９、Ｎｏ．６、Ｊｕｎｅ１９９８））、

・文献３（特開平１０－７８８８０号公報）

等に示されたアーキテクチャが知られている。

【０００５】まず、前記文献２に記載されているＭＵＳＣＡＴアーキテクチャについて説明する。ＭＵＳＣＡＴは、複数のプロセッサエレメントを１チップに集積し、「制御並列」と呼ぶマルチスレッド処理を効率良く実行するアーキテクチャである。各プロセッサエレメントは、それぞれ独立した演算器と独立したレジスタセットを持つ。他方、各プロセッサエレメントはメモリ空間を共有する。

【０００６】ＭＵＳＣＡＴにおける制御並列実行方式について説明する。図１８に示すように、ＭＵＳＣＡＴアーキテクチャのプロセッサは、スレッド生成のためのＦＯＲＫ（フォーク）命令を持ち、この機械命令１つで、隣接プロセッサエレメントに新しいスレッドを生成することができる。

【０００７】ＭＵＳＣＡＴは、１つのスレッドがＦＯＲＫ命令を実行できる回数を高々１回に制限する「ＦＯＲＫ１回モデル」を採用している。各プロセッサエレメントは、単方向のリング状に結合され、あるプロセッサエレメントで実行中のスレッドがＦＯＲＫ命令を実行すると、下流側に隣接するプロセッサエレメントに新しいスレッドが生成される。スレッドがＴＥＲＭ（ターム）命令（スレッド終了命令）を実行すると、自スレッドが終了する。

【０００８】プロセッサエレメント間のデータのやりとりは、レジスタまたはメモリを通じて行われる。ＦＯＲＫ命令を実行すると、当該スレッド（「親スレッド」と呼ぶ）を実行しているプロセッサエレメントのレジスタセットの内容が、生成された新スレッド（「子スレッド」と呼ぶ）を実行するプロセッサエレメントのレジスタセットに論理的にコピーされる。また、親スレッドがＦＯＲＫ命令実行前にメモリにストアした値は、子スレッドがロードして参照することができる。

【０００９】親スレッドが子スレッドを生成した後、親スレッドが子スレッドにデータを渡す唯一の方法は、メモリを介してデータを渡す、というものである。この場合、親スレッドのストアと子スレッドのロードの実行順序を正しいものにするために、一種の同期をとる必要がある。この同期のための措置を、「データ依存保証」と呼ぶ。

【００１０】ＭＵＳＣＡＴは、上記データ依存保証のための方式として、

15

・予め同期を明示的に宣言するＢＬＯＣＫ方式、
・同期無しで実行して、もし同期に失敗したら実行をやり直すＤＳＰ方式、
の２種類のデータ依存保証方式を備える。

【００１１】図１９は、ＢＬＯＣＫ方式の動作を説明する図である。ＢＬＯＣＫ方式によるデータ依存保証方式は、ＭＵＳＣＡＴが備えるＢＬＯＣＫ（ブロック）命令と、ＲＥＬＥＡＳＥ（リリース）命令を用いる方式である。

【００１２】親スレッドは、ストア対象メモリアドレスをＢＬＯＣＫ命令で指定してからＦＯＲＫ命令を実行し、メモリストアを行う。その後、親スレッドは、ＲＥＬＥＡＳＥ命令を実行して、当該メモリアドレスにデータが準備できたことを表明する。親スレッドがＢＬＯＣＫしたアドレスからのロードを子スレッドが実行しようとすると、親スレッドがＲＥＬＥＡＳＥ命令を実行するまで、子スレッドのロード命令は完了しない。このようにして、親スレッドがメモリに書き込んだ値（メモリにストアした値）を、子スレッドが正しく読み出す（メモリからロードする）ことができる。他のデータ依存保証方式であるＤＳＰ方式は、投機実行方式の一つであり、詳細は後述する。

【００１３】次に、図２０を参照して、ＭＵＳＣＡＴアーキテクチャの投機実行機能について説明する。ＭＵＳＣＡＴのＳＰＦＯＲＫ命令はＦＯＲＫ命令の一種であり、制御投機モードのスレッド生成を指示する。ＳＰＦＯＲＫ命令で生成された子スレッドは、いつでも実行を取り消せる状態で実行される。即ち、レジスタ値の変更は許されるが、外部メモリへのアクセスは抑止される。

【００１４】図２０（Ａ）に示すように、親スレッドがＴＨＦＩＸ命令（スレッド確定命令）を実行すると、子スレッドの制御投機モードは解除され、内部で溜っていたメモリ変更操作が実際にメモリに反映される。

【００１５】しかしながら、図２０（Ｂ）に示すように、親スレッドがＴＨＡＢＯＲＴ命令（スレッド破棄命令）を実行すると、子スレッドは破棄され、レジスタやメモリに対して行おうとした変更は全て捨てられる。なお、親スレッドがＴＨＦＩＸ命令、又はＴＨＡＢＯＲＴ命令を実行する前に、ＴＥＲＭ命令により終了すると、子スレッドの制御投機モードは解除される。

【００１６】制御投機モードのスレッドは、結果的に実行することになる可能性が高いが、現時点では実行してもよいか否か判断がつかない場合等に用いられる。即ち、プログラム内の早い段階で制御投機モードスレッドを生成して並列に処理を行い、後になって生成したスレッドでの処理を本当に行うかどうかを判定してスレッドの確定ないし破棄を行う。

【００１７】上記した制御投機モードＦＯＲＫは、制御の流れを予測し、投機的にスレッド生成するものであるが、これとは独立に、ＭＵＳＣＡＴには、データ依存を

16

予測し、投機的にスレッドを実行する機能が用意されている。このようなスレッド状態は、「データ依存投機モード」と呼ばれる。

【００１８】データ依存投機モードを用いる場合、親スレッドは、ＢＬＯＣＫ命令やＲＥＬＥＡＳＥ命令を用いる必要はない。ハードウェアアーキテクチャが、実行時に親子スレッド間のメモリデータ依存（親スレッドのメモリへのストアと子スレッドのメモリからのロードの時間順序）を検査し、子スレッドが誤った値をロードした場合には、子スレッドを再実行させる。

【００１９】前述のＢＬＯＣＫ方式と並ぶ、もう一つのデータ依存保証方式であるＤＳＰ方式は、このデータ依存投機モードを利用する方式である。

【００２０】図２１を参照すると、親スレッドは、データ依存投機モードでＦＯＲＫすることを意味するＤＳＰＩＮ命令を実行した後で、ＦＯＲＫを行う。これによって生成された子スレッドは、データ依存投機モードで実行を開始する。その後、親スレッドがＤＳＰＯＵＴ命令を実行すると、子スレッドはデータ依存投機モードから非データ依存投機モードに戻る。

【００２１】データ依存投機モードは、ＢＬＯＣＫ／ＲＥＬＥＡＳＥ命令が不要であり、データ依存を引き起こすメモリアドレスが事前に分かっていない場合でも、ＦＯＲＫを行える、という長所がある。しかしながら、もし、ＦＯＲＫ後にデータ依存が発生すると、子スレッドは誤ったデータに基づいた部分の実行をやり直すことになる、という短所をもつ。

【００２２】従って、プログラムに内在するデータ依存の状況によって、ＢＬＯＣＫ方式あるいはＤＳＰ方式のうち、適切な方式を選択すべきである。

【００２３】これまで述べたＦＯＲＫ命令や、データ依存保証方式等の制御並列サポート命令により、ＭＵＳＣＡＴアーキテクチャでは、プログラムから並列実行可能な部分をより多く抽出し、プログラムの実行性能を向上させることができる。

【００２４】このようなＭＵＳＣＡＴアーキテクチャ向けのプログラムは、ＦＯＲＫ（フォーク）を行う箇所毎に、前述の命令を用いたコーディングを行わねばならず、このため、高級言語（高水準プログラミング言語）から機械命令列を自動生成するコンパイラが、従来型のアーキテクチャよりも、一層強く求められる。

【００２５】ＭＵＳＣＡＴアーキテクチャ向けコンパイラ技術は、文献４（「制御並列アーキテクチャ向け自動並列化コンパイル手法」（酒井他、情報処理学会論文誌、Ｖｏｌ．４０、Ｎｏ．５、Ｍａｙ１９９９））において開示されている。この文献４に示された技術の概要を以下に説明する。

【００２６】前記文献４には、通常の逐次コンパイラが生成した機械命令列を、ＭＵＳＣＡＴ向け命令列に変換するトランスレータの仕組みが記載されている。与えら

れた機械命令列に対して、制御フロー解析及びデータフロー解析を行い、基本ブロック毎にＦＯＲＫ命令を用いた並列化を試みる。ここで、「基本ブロック」とは、途中で制御の分岐や合流が無い、一続きの命令列のことをいう。

【００２７】基本ブロック毎の並列化は、まず、基本ブロック末尾の分岐命令を、制御投機モードＦＯＲＫ命令（ＳＰＦＯＲＫ命令）に置き換えることから始まる。これは、ある基本ブロックに後続する複数の基本ブロックの一方を、当該基本ブロックと並列に実行しようとすることを意味する。後続基本ブロックのうちいずれを選択するかは、前記文献４では、ループ構造の場合は、ループの戻り方向がよいとされている他、プロファイル情報を用いて分岐確率の高い方を選択することが示されている。

【００２８】次に、当該基本ブロックと、ＦＯＲＫ先基本ブロック以降との間でのレジスタ、及びメモリアクセスによるデータ依存を調査する。

【００２９】そして、当該基本ブロック内で、ＦＯＲＫ命令ができるだけ上流に位置するように、基本ブロック内で命令の並べ換えを行う。命令並べ換えに際しては、データ依存関係に留意し、レジスタを介して正依存がある場合には、その命令をＦＯＲＫ命令よりも上流側に配置する。メモリを介して正依存がある場合には、ＤＳＰＩＮ命令ないし依存メモリアドレスを引数としたＢＬＯＣＫ命令を、ＦＯＲＫ命令の直前に新たに挿入する。

【００３０】両者の使い分けについて、前記文献４では、依存をひき起こすメモリアドレスがストア直前まで決定できない場合、及び、依存をひき起こすメモリアドレスが決定できた場合でもメモリアドレス値がある数より多い場合には、ＤＳＰＩＮ命令、それ以外の場合には、ＢＬＯＣＫ命令を用いるものとされている。なお、前記文献４では、この他にも、ＭＵＳＣＡＴ特有の命令生成手順が述べられているが、本発明の主題とは直接の関連性は薄いため、ここでは、あらためて説明は行わない。

【００３１】

【発明が解決しようとする課題】ところで、上記した従来の技術は、下記記載の問題点を有している。

【００３２】第１の問題点は、ＦＯＲＫ先を選択する基準に、まだ改良の余地がある、ことである。

【００３３】上記した従来の技術では、静的にはループ構造の把握、動的には実行履歴のプロファイル情報を得て、それに基づいてＦＯＲＫ先を決定している。しかしながら、これらは、制御の流れ上、実行される確率が高い部分をＦＯＲＫしているにすぎず、データ依存性については、考慮されていない。このため、過度のデータ依存が存在する部分をマルチスレッド実行しようとしても、並列実行区間が短くなったり、あるいはデータ依存箇所で待ち合わせが起こるため、結果として、得られる

性能向上は、小さなものとなってしまう、という問題がある。

【００３４】第２の問題点は、データ依存対処手順がレジスタとメモリとで異なっている、ということである。

【００３５】前記文献４等に記載されている手順では、レジスタを介した依存と、メモリを介した依存とで、命令の並べ換えや、新規挿入命令の処理が異なっている。しかしながら、コンパイラ内部では、「中間項」と呼ばれる仮想的な変数を用いた表現が用いられることが一般に行われており、その段階で、レジスタかメモリかの判断に基づく並列化処理を行うことは困難である。

【００３６】他方、一般的なコンパイラ内部でも、最終段近くになれば、レジスタ割り当てが済み、レジスタとメモリの区別が確定しているが、この段階で並列化を行おうとすると、既に、最適化された制御フローやデータフローに悪影響を与えないように、プログラム構造を変更する処理が困難になる。

【００３７】したがって、本発明は、上記問題点に鑑みてなされたものであって、その目的は、コンパイラ内部の中間項レベルで的確な判断により並列化処理を行えるようにし、もってマルチスレッドマイクロプロセッサ向けプログラム変換装置及び方法並びに記録媒体を提供することである。

【００３８】本発明の他の目的は、並列実行時により一層の並列性能を引き出すような目的コードを生成できるプログラム変換装置（コンパイラ）及び方法並びに記録媒体を提供することにある。これ以外の本発明の目的、特徴、利点等は、下記の実施の形態の記載等から、当業者には直ちに明らかとされるであろう。

【００３９】

【課題を解決するための手段】前記目的を達成する本発明の第１のプログラム変換装置は、中間プログラム上でレジスタ割り当てを試行して割り当て情報を先行取得するレジスタ割り当てを試行部と、メモリを介して発生するデータ依存の距離の計算部と、を備え、メモリを介するデータ依存の距離を考慮してＦＯＲＫ先を決定し、条件分岐をＦＯＲＫ命令に置き換え、レジスタ割り当て試行結果を参照して、中間プログラム上で、ＦＯＲＫ命令前後の命令並べ換えを行う。

【００４０】また、本発明の第２のプログラム変換装置は、前記第１のプログラム変換装置に加えて、プロファイル情報から条件分岐確率及びデータ依存発生頻度を計算する部分とを備え、データ依存の距離と条件分岐確率とデータ依存発生頻度とデータ依存メモリ箇所数からＦＯＲＫ先を決定し、条件分岐をＦＯＲＫ命令に置き換え、レジスタ割り当て試行結果を参照して中間プログラム上でＦＯＲＫ命令前後の命令並べ換えを行う。

【００４１】本発明の並列化方法は、原始プログラムをコンパイルしてマルチスレッド型のターゲットプロセッサ装置向けの目的プログラムを生成出力するコンパイラ

における、構文解析の結果出力される中間プログラムに対して並列化を含む最適化処理を行う方法であって、
（ａ）並列化に先立ってレジスタ割り当てを試み、中間プログラム上の各変数、及び中間項のレジスタ割り当て状況を予測するレジスタ割り当て試行ステップと、
（ｂ）前記レジスタ割り当て試行結果に基づいて、前記中間プログラムにおける条件分岐部分をスレッド生成命令を用いた並列コードに変換するか否かの決定を行うか、もしくは、並列コードに変換するか否かの決定と並列コードに変換する場合その並列実行方式の決定を行うフォーク（ＦＯＲＫ）箇所決定ステップと、（ｃ）前記フォーク（ＦＯＲＫ）箇所決定部での決定結果に基づいて、前記中間プログラム中の条件分岐部分を、スレッド生成命令を用いた並列コードに変換し、前記レジスタ割り当て試行結果を参照して、スレッド間のメモリを介したデータ依存関係を保証する命令を、前記スレッド生成命令の前後に挿入すると共に、スレッド生成が早い段階で行われるように、前記スレッド生成命令の前後の命令を並べ換える命令並べ換えステップと、（ｄ）並列化され並べ換えられた命令列に対して、物理レジスタが割り当てられるか否かに関して、前記レジスタ割り当ての試行時と同じ割り当て結果となるように確定的なレジスタ割り当てを行うレジスタ割り当てステップと、を含む。

【００４２】本発明において、ターゲットプロセッサ装置は、好ましくは、複数のプログラムカウンタと、複数のスレッド実行装置と、を備え、前記複数のスレッド実行装置は、前記複数のプログラムカウンタに従って、複数のスレッドの命令を、同時に、フェッチ、解釈、実行し、スレッド生成時以降に、レジスタセットに及ぼした変更を、後に取り消し可能な制御投機的モードでスレッドを実行すること、及び、自スレッドがメモリ領域から値をロードした後に、自スレッドを生成した親スレッドが同一メモリ領域に値をストアした場合に、自スレッドの少なくとも当該ロード以降の処理結果を破棄し、それらの処理を再実行するデータ依存投機的モードでスレッドを実行すること、が可能とされ、命令セットとして、
・前記スレッド実行装置で実行中のスレッドが制御投機的モードの新たなスレッドを生成すること、
・指定された条件が成立していれば自スレッドを終了させると共に、自スレッドが生成した制御投機的モードのスレッドの制御投機的モードを解除すること、
・生成した制御投機的モードのスレッドを破棄すること、
・自スレッドが生成するスレッドが指定されたアドレスのメモリ領域からのロードを行う際に、その動作を一時停止させることをあらかじめ指示すること、
・指定されたメモリアドレスに対する前記ロード一時停止指示を解除すること、
・前記スレッド実行装置で実行中のスレッドがデータ依存投機的モードの新たなスレッドを生成すること、及び、
・自スレッドが生成したデータ依存投機的モードのスレッドのデータ依存投機的モードを解除すること、
が、単一又は高々数個の機械語命令の組合せで実行できる命令セットを有するマルチスレッドプロセッサよりなる。

【００４３】
【発明の実施の形態】次に、本発明の実施の形態について図面を参照して説明する。

【００４４】［実施の形態１］図１は、本発明の一実施の形態の構成の一例を示す図である。図１を参照すると、プログラム変換装置（コンパイラ）２は、原始プログラム（ソースプログラム）１を読み込んで並列化を含むコンパイル処理を行い、その結果の目的プログラム（ターゲットプログラム）３を出力する。

【００４５】目的プログラム実行装置４は、目的プログラム３を入力して、目的アーキテクチャの命令実行を行うと共に、当該プログラムの実行情報を収集し、プロファイル情報ファイル５を出力する。

【００４６】図１を参照すると、プログラム変換装置２は、
・入力した原始プログラム１の文法（シンタックス）を解釈して、構文解析し中間プログラムを生成する構文解析装置１０と、
・構文解析装置１０から受け取った中間プログラムに対して並列化を含む最適化処理を行う並列化装置１１と、
・並列化装置１１から受け取った最適化済みの中間プログラムから目的アーキテクチャ用命令列を生成するコード生成装置１２と、
を備えている。

【００４７】並列化装置１１は、補助情報として、プロファイル情報ファイル５を入力としてより高度な最適化処理を行うことができる。

【００４８】なお、本発明において、構文解析装置１０は、周知の構成が用いられ、例えば字句解析（レキシカルアナリシス）、構文解析（パージング）、意味解析（セマンティックアナリシス）の各解析部からなる構成としてもよいことは勿論である。また最適化済みの中間プログラムから目的アーキテクチャ用命令列を生成するコード生成装置１２も、周知のものが用いられる。

【００４９】図２は、本発明の一実施の形態のプログラム変換装置２における並列化装置１１の構成を示す図である。図１及び図２を参照して、本発明の要部をなす並列化装置１１について詳細に説明する。

【００５０】並列化装置１１は、構文解析装置１０が生成した中間プログラム６を受け取り、並列化を含む最適化処理を行い、後段のコード生成装置１２に中間プログラム７を渡す。

【００５１】並列化装置１１は、
・入力となる中間プログラム６を読み込んで制御フロー

やデータフローを解析する中間プログラム入力部２０と、

・制御フロー、データフロー及びプロファイル情報をもとに並列化する箇所を決定するＦＯＲＫ箇所決定部２１と、

・中間プログラム上の中間項に対してレジスタ割り当てを試行するか、あるいは実行するレジスタ割り当て部２２と、

・決定された並列化箇所とデータフロー等の情報から並列化箇所前後の命令の並べ換えを行う命令並べ換え部２３と、

・並列化を含む様々な変換を終えた命令列を再度中間プログラム形式で出力する中間プログラム出力部２４と、

・一度目的プログラムを実行して得たプロファイル情報ファイル５を入力して内部形式に変換するプロファイル情報入力部２５と、

を備えている。

【００５２】次に、図３、図４、及び図５のフローチャートを参照して、本発明の一実施の形態の並列化動作について、詳細に説明する。

【００５３】本発明の一実施の形態においては、中間プログラム上で、次の制御並列関連命令が記述できる。

【００５４】（１）制御投機ＦＯＲＫ命令：制御投機ＦＯＲＫ命令は、オペランドで指し示された命令から実行を開始する制御投機モードの子スレッドを生成し、自スレッドは後続命令の実行を続ける命令である。

【００５５】（２）スレッド終了命令：スレッド終了命令は、自スレッドを終了させるとともに子スレッドを確定モードに移行させる命令である。

【００５６】（３）子スレッド破棄命令：子スレッド破棄命令は、制御投機モードの子スレッドを破棄する命令である。

【００５７】（４）ブロック設定命令：ブロック設定命令は、オペランドで指し示されたメモリアドレスにブロックを設定し、それによって子スレッドが当該メモリアドレスからのロードを実行しようとした際に子スレッドの実行を一時停止（ブロック）させるように、プロセッサに指示する命令である。

【００５８】（５）ブロック解除命令：ブロック解除命令は、オペランドで指し示されたメモリアドレスに設定されているブロックを解除するとともに、当該メモリアドレスからのロードで一時停止（ブロック）されている子スレッドの実行を再開させるように、プロセッサに指示する命令である。

【００５９】（６）レジスタ割り当て指示命令：レジスタ割り当て指示命令は、オペランドで指定した中間項ないし変数に物理レジスタを割り当てるように、あるいは逆に、指定した中間項ないし変数にメモリ上の領域を割り当てるように、レジスタ割り当て部に指示する命令である。

【００６０】図３は、本発明の一実施の形態のＦＯＲＫ箇所決定部２１の動作概要を示す図である。ＦＯＲＫ箇所決定部２１は、与えられた中間プログラム中の各関数単位で、図３に示された動作を行う。即ち、

・ステップ２５にて、当該関数内でレジスタ割り当てを試行し、その後、

・ステップ２６にて、当該関数内に含まれる各条件分岐命令に対してＦＯＲＫ箇所決定処理を行う。

【００６１】このステップ２６の段階では、並列化を含む各種最適化処理は未完了であり、まだ確定的なレジスタ割り当て処理を行う段階ではない。そこで、レジスタ割り当ての途中までを実行し、中間プログラム中のどの中間項／変数にレジスタが割り当てられ、どの中間項／変数がメモリ上に配置されるのか、という情報を得た段階で、実際に割り当ては行わずにレジスタ割り当て処理を中止する。

【００６２】レジスタ割り当ての試行は、後述するように、レジスタ割り当て部２２（図２参照）で行う。レジスタ割り当て部２２は、どの中間項／変数に、どのレジスタを割り当てるかを決めた段階で処理を終え、中間項／変数を実際のレジスタに置き換えることなく、レジスタ割り当て状況のみを返す仕組み（機構）を備える。

【００６３】図４は、各条件分岐命令に対するＦＯＲＫ箇所決定部２１の処理（図３のステップ２６）の詳細を示す図である。

【００６４】図４を参照すると、ステップ３１では、当該条件分岐命令が、入力中間プログラム中のループ構造（繰り返し構造）の戻り枝に相当する分岐命令か否かを判定する。ループ構造の検出方法については、例えば、文献５（「コンパイラＩＩ原理・技法・ツール」、Ａ．Ｖ．エイホ他著、原田賢一訳、サイエンス社、１９９０）の第７３４頁～第７３７頁の記載が参照される。

【００６５】ステップ３１の判定で、当該条件分岐命令が、ループ戻り枝に相当する分岐命令である場合、ステップ３５にて、当該戻り枝方向、即ちループ継続方向をＦＯＲＫ先として、この条件分岐箇所を、ＦＯＲＫ箇所に選定する。その理由は、一般に、ループ構造は、複数回繰り返し実行される傾向にあるため、ループ戻り枝に相当する分岐命令では、戻り枝側に分岐する確率が高い、からである。

【００６６】ステップ３１において、戻り枝分岐ではない場合、ステップ３２において、条件分岐の２つの分岐先の各々に対してデータ依存の距離の最小値を計算する。

【００６７】ここで、「データ依存の距離」とは、現在、処理対象としている基本ブロック内で定義され、その分岐先で参照される可能性がある中間項及び変数のうち、ステップ２６により、メモリ上に配置されると見込まれる、中間項／変数各々に対して、その分岐先基本ブロックの中で、該メモリ参照命令が先頭から、どれぐら

いの位置にあるかを、中間プログラム中のステップ数で表したものである。

【００６８】ステップ３２では、２つの分岐先について、各中間項／変数のデータ依存の距離の最小値を計算する。

【００６９】次のステップ３３では、ステップ３２で条件分岐の両側に関して求めた２つのデータ依存の距離の最小値を比較し、両者に一定以上の差があるか否かを判定する。

【００７０】ステップ３３において、一定以上の差がある場合、ステップ３６にて、データ依存の距離の最小値の大きい側の分岐方向をＦＯＲＫ先とし、この条件分岐箇所を、ＦＯＲＫ箇所に選定する。その理由は、ここで選定した分岐方向を新スレッドとしてＦＯＲＫする方がもう一方をＦＯＲＫする場合に比べ、ＦＯＲＫ直後に、データ依存で停止してしまう可能性が低い、からである。

【００７１】ステップ３２で、データ依存の距離を求める際、単純な命令ステップ数では、なく、各命令が目的アーキテクチャのプロセッサ上で実行される際に要するものと推定されるサイクル数（クロックサイクル等）を用いることもできる。これにより、データ依存が発生する時期の遅い方を選択する、という、上記ステップ３３の選択がより的確なものになる。

【００７２】ステップ３４では、その直前のステップ３３で、データ依存の距離の最小値に一定以上の差がない場合、元の中間プログラム内で分岐先であった側（分岐命令のｔａｋｅｎ側（分岐成立側））をＦＯＲＫ先とし、この条件分岐箇所を、ＦＯＲＫ候補に選定する。このＦＯＲＫ先の選定は、従来の技術として挙げた上記文献等に示されているものと同じ理由による。

【００７３】以上のようにして、ＦＯＲＫ箇所決定部２１は、ＦＯＲＫ箇所と、ＦＯＲＫ先を決定する。

【００７４】本発明の一実施の形態では、図４に詳細を示した、ＦＯＲＫ箇所決定部２１の処理によって、通常は、最適化の最終段で行うレジスタ割り当て処理を待たずに、レジスタ割り当て情報を参照して中間プログラム上で並列化処理を行うことが可能となる。

【００７５】また、本発明の一実施の形態においては、図４のステップ３２、ステップ３３の処理を設けることで、ループ構造以外の部分での性能向上の可能性が高まる。

【００７６】図５は、本発明の一実施の形態の並列化装置１１の命令並べ換え部２３の動作を説明するための流れ図である。また、図６は、命令並べ換え部２３の命令並べ換え処理の説明を補足する図である。以下、図５及び図６を参照しながら、命令並べ換え部２３の動作を説明する。

【００７７】図５を参照すると、命令並べ換え部２３は、ＦＯＲＫ箇所決定部２１が決定したＦＯＲＫ箇所を

含む基本ブロック毎に、ステップ４０からステップ４４の一連の処理を行う。なお、ステップ４０からステップ４４の処理は、すべて中間プログラムに対して行う。これらのステップの説明に現れる命令は、すべて中間プログラム上の対応する命令を指す。

【００７８】図６（Ａ）は、命令並べ換え部２３が処理対象とする基本ブロックの構造を模式的に示した図である。図６（Ａ）における「Ｍ：＝‥」はメモリへのストアを行う命令を意味する。

【００７９】まず、図５のステップ４０にて、中間プログラム中の各中間項及び変数がレジスタに対応付けられるかメモリに対応付けられるかを調べる。これは、図４のステップ３２と同様に、レジスタ割り当てを途中まで試行することで判定する。ステップ３２からこのステップ４０までの処理では、ＦＯＲＫ先を決定するのみで、中間プログラム自体は変更されず、レジスタ割り当て状況に差が出ないので、ステップ３２のときのレジスタ割り当て試行結果を保存しておき、その情報をステップ４０で参照するのがよい。

【００８０】次のステップ４１では、現在処理対象としている基本ブロックの末尾にある分岐命令を、制御投機モードＦＯＲＫ命令に置換（変換）する。制御投機ＦＯＲＫ命令のオペランド、即ちＦＯＲＫ先は、ＦＯＲＫ箇所決定部２１が選択したＦＯＲＫ先とする。図６（Ｂ）は、ステップ４１までの処理を終えた段階の基本ブロック構造を示す図である。

【００８１】ステップ４２では、中間プログラム中で、制御投機ＦＯＲＫ命令の直前にある分岐条件計算文（分岐条件式）を、制御投機ＦＯＲＫ命令直後に移動させると共に、その移動先の直後、即ち当該基本ブロックの末尾に、「分岐条件成立時には自スレッドを終了させて子スレッドを確定モード（非制御投機モード）に移行させ、分岐条件非成立時には、子スレッドを破棄（アボート）させて自スレッドが後続命令列の実行を続ける」というような一群の命令列を挿入する。図６（Ｃ）は、ステップ４２までの処理を終えた段階の基本ブロック構造を示す図である。

【００８２】ステップ４３では、現在処理対象としている基本ブロックにおいて、ＦＯＲＫ命令よりも手前、即ち、上流側にある文のうち、ステップ４０において、メモリに対応付けられた中間項及び変数への代入となる文の各々について、その代入文を、ＦＯＲＫ命令よりも後方、即ち、下流側へ移動させると共に、ＦＯＲＫ命令の直前に、ブロック設定命令を挿入し、代入文（：＝）の移動先の直後に、ブロック解除命令を挿入する。ここで、挿入するブロック設定命令、及びブロック解除命令のオペランドは、移動させた代入文の代入先中間項／変数を表すメモリアドレスとする。

【００８３】中間プログラムの形態によっては、この段階で、具体的なメモリアドレスが確定していない場合が

あるが、その場合は、中間プログラム中での中間項／変数表現と同様の形式を用いてブロック設定命令及びブロック解除命令を表現し、後に、中間プログラムからコード生成を行う際に実効メモリアドレスを示す命令列に変換すればよい。

【００８４】ステップ４３では、ステップ４０にてレジスタに対応付けられた中間項及び変数への代入となる文のうち、ＦＯＲＫ先で参照される可能性がある中間項／変数への代入となる文は、ＦＯＲＫ命令後に移動させてはならない。その理由は、レジスタの値はＦＯＲＫ時点で子スレッドに継承されるため、ＦＯＲＫ後に親スレッドが定義したレジスタ値は子スレッド側に渡されないためである。

【００８５】このように、必ずしも全ての文をＦＯＲＫ命令後に移動できるわけではないため、ステップ４３において、メモリへの代入文をＦＯＲＫ後に移動させる際には、移動させようとしている文と、その後続命令との間のデータ依存関係を調べ、移動によって実行順序が入れ替わっても、移動前と同じ演算結果が得られる場合に限って移動を行わなければならない。

【００８６】なお、このとき必要となるデータ依存関係は、コンパイラ技術として一般的なものでよく、本発明の一実施の形態では、図２における中間プログラム入力部２０が作成した制御フロー及びデータフロー解析結果を基に、調査して得ることができる。

【００８７】図６（Ｄ）は、ステップ４３までの処理を終えた段階の基本ブロック構造を示す図である。

【００８８】図５のステップ４４では、これまでのステップで各中間項及び変数のレジスタ割り当てについて仮定した情報を、現在処理対象の基本ブロックの先頭に挿入する。この情報は、実際の機械命令に対応するものではないので、中間プログラム上の疑似命令であるレジスタ割り当て指示命令を用いて記述する。

【００８９】図６（Ｅ）は、上記ステップ４４までの処理を終えた段階の基本ブロック構造を示す図である。

【００９０】本発明の一実施の形態において、ステップ４０及びステップ４４を具備することを特徴の一つとしている。即ち、・ステップ４１のＦＯＲＫ変換処理に先立って、レジスタ割り当て状況を調べるステップ４０と、・ステップ４１のＦＯＲＫ変換処理で仮定したレジスタ割り当て状況を、レジスタ割り当て部２２に指示するステップ４４と、を有しており、このため、中間項及び原始プログラム中の変数表現を用いた中間プログラムに対しても、詳細な命令並べ換えを行うことができる。

【００９１】なお、ステップ４１からステップ４３の処理は、上記文献４（「制御並列アーキテクチャ向け自動並列化コンパイル手法」（酒井他、情報処理学会論文誌、Ｖｏｌ．４０、Ｎｏ．５、Ｍａｙ１９９９））の第２０４９頁～第２０５０頁に記載されているものと同様のものである。

【００９２】図２に戻ると、レジスタ割り当て部２２は、中間プログラム中の中間項に、目的アーキテクチャが有する物理レジスタあるいはメモリ上に割り当てられた領域を対応付けるレジスタ割り当て処理を行う。

【００９３】レジスタ割り当ての基本的な方法としては、中間項の使用頻度の高いものから割り当てる方法や干渉グラフの彩色（カラーリング）による割り当て方法などを用いることができる。これらの割り当て方法は、例えば上記文献５（「コンパイラＩＩ　原理・技法・ツール」、Ａ．Ｖ．エイホ他著、原田賢一訳、サイエンス社、１９９０」）の第６５９頁～第６６５頁の記載が参照される。

【００９４】レジスタ割り当て部２２におけるレジスタ割り当て処理は、図５のステップ４４で挿入したレジスタ割り当て指示命令に従う点で、通常のコンパイラでのレジスタ割り当て処理とは異なる。レジスタ割り当て方法として、例えば中間項使用頻度順のレジスタ割り当て方法を採る場合は、レジスタ割り当て指示命令でレジスタに乗せるよう指定された各中間項の頻度を、他の中間項よりも高く設定することで、指示された中間項が物理レジスタに割り当てられる可能性を高める。

【００９５】また、レジスタ割り当て方法として、カラーリングによる方法を採る場合は、干渉グラフ中で割り当て可能な物理レジスタ総数を越える隣接節点数を持つ節点に対応する中間項の中からメモリ上に割り付ける中間項をいくつか選択する際、図５のステップ４４で挿入したレジスタ割り当て指示命令でレジスタに乗せるように、指示されていない中間項を優先的に選択することで、レジスタ割り当て指示された中間項が物理レジスタに割り当てられる可能性を高める。

【００９６】レジスタ割り当て部２２にて、もし、レジスタ割り当て指示命令でレジスタに乗せるように指示された全ての中間項に物理レジスタを割り付けられなかった場合、それらの中間項は、メモリ領域に割り当てられるようにする。その理由は、命令並べ換え部２３によって、レジスタ割り当てを前提としている中間項はＦＯＲＫ命令より手前（上流側）で値が確定するようにプログラムが変換されているため、それらの中間項が、たとえメモリ上に割り当てられたとしてもプログラムの実行結果は、変化しないからである。

【００９７】逆に、レジスタ割り当て部２２は、レジスタ割り当て指示命令でレジスタに乗せるよう指示されていない中間項をレジスタに割り当てないように、レジスタ割り当て処理を行う。これは、従来からのレジスタ割り当て処理における、アドレスを参照されている変数や、ｖｏｌａｔｉｌｅ変数等のレジスタ上に配置できない変数に対する処理と同様の枠組で対処できる。

【００９８】

【実施例】次に、上記した本発明の一実施の形態の動作についてさらに詳細に説明すべく、具体的な実施例を用

いて説明する。

【００９９】［実施例１］図８は、本発明の一実施例として、並列化装置１１によって並列化される中間プログラムの一例を示す図である。ここでｔ１～ｔ２８は中間項であり、Ｉ、Ｊ、Ｋ、Ｒ、Ｘは原始プログラム中で宣言された変数である。

【０１００】「：＝」は、右辺の値を左辺で示す場所に格納することを意味する。

【０１０１】「＆」は、変数の置かれるメモリアドレスを返す前置演算子である。

【０１０２】「ｍｅｍ（ａ）」は、ａの値をアドレスとするメモリの内容を示し、「：＝」の右辺ならメモリロード、左辺ならメモリストアを意味する。

【０１０３】Ｌ１、Ｌ２、Ｌ３はラベルである。

【０１０４】図８の左端（の列）に示されている括弧付きの数字（１）～（３７）は、説明の都合により付した番号であり、右端の（Ｂ１）～（Ｂ３）は基本ブロック番号を示す。

【０１０５】図７は、本実施例の動作の説明をする際に用いる中間プログラム上の制御並列関連命令一覧を示す図である。

【０１０６】・ＳＰＦＯＲＫ　ｌ　は、オペランドｌから実行開始する投機（制御）モード子スレッドを生成する。

・ＴＴＥＲＭ　ｃ　は、オペランドｃが真の場合、自スレッドを終了し、子スレッドを確定する。

・ＦＴＥＲＭ　ｃ　は、オペランドｃが偽の場合、自スレッドを終了し、子スレッドを確定する。

・ＴＨＡＢＯＲＴ　は、投機モードの子スレッドを破棄する。

・ＢＬＯＣＫ　ｍ　は、オペランドｍで指定したメモリアドレスをブロック指定する。

・ＲＥＬＥＡＳＥ　ｍ　は、オペランドｍで指定したメモリアドレスに設定したブロックを解除する。

・ＤＳＰＩＮ　は、後続のＦＯＲＫで生成した子スレッドをデータ依存投機モードで生成する。

・ＤＳＰＯＵＴ　は、子スレッドのデータ依存投機モードを解除する。ＲＤＣＬ　ｔ１，…　は、オペランドｔ１，…で指定した中間項／変数をレジスタに割り当てるように指示する。ＭＤＣＬ　ｔ１，…　は、オペランドｔ１，…で指定した中間項／変数をメモリに割り当てるように指示する。

【０１０７】図２を参照すると、ＦＯＲＫ箇所決定部２１が、図８に示された中間プログラムを受け取り、ＦＯＲＫ箇所の決定を行う。

【０１０８】図３のステップ２５にて、レジスタ割り当てが試行され、ｔ１～ｔ２８の全ての中間項、及び変数Ｉ、Ｋ、Ｒには物理レジスタが割り当てられたが、変数Ｊ、Ｘには、メモリ上の領域が割り当てられた、という試行結果を得たとする。

【０１０９】図３のステップ２６では、条件分岐命令を含む基本ブロック（Ｂ１）が並列化変換の対象となる。以下、図４を参照して、ＦＯＲＫ箇所決定処理部２７の動作を説明する。

【０１１０】ステップ３１で、ループ構造チェックを行うが、図８の（１１）の条件分岐は、ループ戻り枝ではないため、ステップ３２に進む。

【０１１１】ステップ３２では、図８の基本ブロック（Ｂ１）と（Ｂ２）、（Ｂ１）と（Ｂ３）の間でのデータ依存の距離を求める。

【０１１２】まず（Ｂ１）－（Ｂ２）間のデータ依存を見てみると、（Ｂ１）の（７）におけるメモリストアと、（Ｂ２）の（１４）、（１８）、（１９）におけるメモリロードとの間にデータ依存がある。

【０１１３】これらのメモリアクセスは、実際には、配列要素Ｘ［Ｉ］へのストアと、変数Ｊ、配列要素Ｘ［Ｊ］からのロードであり、配列Ｘと、変数Ｊのメモリ領域が重ならず、ＩとＪの値が異なっていれば、データ依存は発生しない。すなわちブロック（Ｂ１）において、（１１）のｔ１：＝＆Ｘは、配列Ｘの先頭アドレスを中間項ｔ１に格納し、１配列要素は４バイトであり（ｔ４＝ｔ２＊ｔ３）、ｔ５＝ｔ１＋ｔ４により、（７）のｍｅｍ（ｔ５）が、Ｘ［Ｉ］のメモリアドレスを示し、このアドレスへのｔ６のメモリストアであり、またブロック（Ｂ２）において、（１８）のｔ１５：＝ｍｅｍ（ｔ１４）は、Ｘ［Ｊ］からのメモリロードを表している。

【０１１４】しかし、ここでは、データ依存解析の結果、このような条件が常に満たされる確証を得られず、メモリストアとメモリロードとの間に、潜在的なメモリデータ依存が存在すると判断されたものとして、説明を続ける。

【０１１５】（Ｂ１）－（Ｂ２）間の各データ依存の距離は、図８の中間プログラム上のステップで数えて、各々１、５、６である。ここで、（１２）にあるラベルＬ１は、非実行文であるためステップ数に含めず、（１３）にある命令を０、（１４）にある命令を１、（１５）にある命令を２、というように数えている。これにより、（Ｂ１）－（Ｂ２）間のデータ依存の距離の最小値は１である。

【０１１６】同様に、（Ｂ１）－（Ｂ３）間では、（Ｂ１）の（７）と、（Ｂ３）の（２９）、（３３）との間にデータ依存があり、その距離は、各々５、９である。つまり、（Ｂ１）－（Ｂ３）間のデータ依存の距離の最小値は５である。

【０１１７】ステップ３３では、上で求めたデータ依存の距離の最小値である６と１０とを比較し、その差が十分であると判断し、データ依存の距離の最小値の大きい側である（Ｂ３）側の分岐方向を、ＦＯＲＫ先に選定する。

【０１１８】もし、図３のステップ２５において、変数Ｊがメモリではなくレジスタに割り当てられるという見通しを得ていたすると、（Ｂ１）－（Ｂ２）、（Ｂ１）－（Ｂ３）のデータ依存の距離の最小値は各々５、９となり、ステップ３３にて（Ｂ２）側をＦＯＲＫ先に選定することになる。

【０１１９】次に、図５を参照して、命令並べ換え部２３の動作を具体的に説明する。本実施例では、中間プログラム上で、図７に示す制御並列関連命令が利用できるものとする。

【０１２０】先の説明で例に挙げた中間プログラムを与えると、命令並べ換え部２３は、基本ブロック（Ｂ１）から（Ｂ３）へのＦＯＲＫを行うように、命令並べ換え処理を行う。

【０１２１】図５のステップ４０では、中間項ｔ１～ｔ２８と、変数Ｉ、Ｋ、Ｒがレジスタに割り当てられる、という情報を取得する。

【０１２２】図５のステップ４１では、図８における（１１）の分岐命令（ｔｈｅｎ　以下のｇｏｔｏ　Ｌ２）を、制御投機ＦＯＲＫ命令であるＳＰＦＯＲＫ命令に置き換える。ＳＰＦＯＲＫ命令のオペランドは、ＦＯＲＫ先として決定した（Ｂ３）、即ちラベルＬ２とする（図９の（５８）のＳＰＦＯＲＫ　Ｌ２）。

【０１２３】図５のステップ４２では、図８における、分岐条件を計算する文を構成する一連の命令（８）～（１０）を、ＳＰＦＯＲＫ命令の直後に移動させ、（１１）の分岐条件に合わせた条件付きスレッド終了命令である、ＦＴＥＲＭ命令（条件不成立時に自スレッドを終了する命令）、更には、ＴＨＡＢＯＲＴ命令と、後続ブロックである（Ｂ１）への無条件分岐命令（ｇｏｔｏ　Ｌ１）を挿入する。

【０１２４】これらの命令列は、以下のように動作するように挿入する。

【０１２５】即ち、ＳＰＦＯＲＫ後に、分岐条件計算を行って、ＳＰＦＯＲＫが本当に正しかったか否かを判定する。

【０１２６】ＳＰＦＯＲＫが正しい（投機成功）ならば、条件付きスレッド終了命令の一種ＦＴＥＲＭにより、自スレッド（親スレッド）を終了させると同時に、生成した子スレッドを、制御投機モードから確定モード（非制御投機モード）に変更する。

【０１２７】一方、ＳＰＦＯＲＫが正しくなかった（投機失敗）ならば、ＦＴＥＲＭ命令は何もせず、後続のＴＨＡＢＯＲＴ命令によって、子スレッドを破棄し、無条件分岐命令（図９の（６４）のｇｏｔｏ　Ｌ１）によって、自スレッドは、ＦＯＲＫ先とは逆の分岐先の命令の実行を続けるようにする。図９は、図８に示す中間プログラムに対して、図５のステップ４２までの変換を施した中間プログラムを示す図である。

【０１２８】図５のステップ４３では、メモリストア文を構成する一連の命令である図９の（５６）と（５７）を、ＳＰＦＯＲＫ命令（５８）の直後に移動し（図１０の（１０８）、（１０９））、その直後に、ＲＥＬＥＡＳＥ命令を挿入する（図１０の（１１０））。

【０１２９】ＲＥＬＥＡＳＥ命令のオペランドは、移動させたストア命令（５７）のオペランドと同じ中間項を用いる。またＳＰＦＯＲＫ命令（５８）の直前に、ストア命令（５７）のオペランドと同じ中間項をオペランドとするＢＬＯＣＫ命令を挿入する。図１０は、ここまでの変換を施した中間プログラムである。

【０１３０】ここで、メモリストア文をＦＯＲＫ後に移動させる際、命令（５１）～（５７）全部ではなく、（５６）と（５７）のみを移動させた理由は、実効メモリアドレスを保持する中間項を、ＢＬＯＣＫ命令及びＲＥＬＥＡＳＥ命令で再利用できるようにするためである。コンパイラ分野において、「共通部分式削除」と呼ばれる最適化機能を用いれば、ここでＢＬＯＣＫ及びＲＥＬＥＡＳＥのたびに実効アドレスを再計算するようなプログラム変換を施しても、共通部分式削除最適化により冗長な実効アドレス計算は削除される。

【０１３１】図５のステップ４４では、各基本ブロック（Ｂ１）、（Ｂ２）、（Ｂ３）の先頭に、ＲＤＣＬ命令及びＭＤＣＬ命令を挿入する。これらの命令は、後段のレジスタ割り当て部２２への指示であり、このうち、・ＲＤＣＬ命令は、指定した変数ないし中間項へのレジスタ割り当てを行い、・ＭＤＣＬ命令は、指定した変数ないし中間項へのメモリ領域の割り当てを行う。

【０１３２】図１１は、ここまでの変換を施した中間プログラムである。（２０１）～（２０３）、（２２１）～（２２３）、（２３５）～（２３７）がステップ４４で挿入した命令である。

【０１３３】再度、図２を参照すると、命令並べ換え後、レジスタ割り当て部２２が、レジスタに置くべき変数や中間項に物理レジスタを割り当てる。

【０１３４】本実施例では、プログラム変換装置から出力される目的プログラムが実行される装置のアーキテクチャ（ターゲットアーキテクチャ）は、ｒ０～ｒ３１の３２本の物理レジスタを持ち、レジスタ変数（原始プログラム中の変数のうち、物理レジスタに割り当てられるもの）用に、物理レジスタｒ１０～ｒ１９、作業用レジスタ（主として演算途中結果を保持するレジスタ）用に、物理レジスタｒ２０～ｒ３０を用いるようなレジスタ割り当てを行うものとする。

【０１３５】レジスタ割り当て部２２は、
・変数Ｉ、Ｋ、Ｒに、それぞれｒ１１、ｒ１３、ｒ１２を割り当て、
・中間項ｔ１～ｔ２８には、ｒ２０～ｒ３０を順に割り当てる。

【０１３６】図１２は、図１１に示した中間プログラムに対して、レジスタを割り当てた後の中間プログラムを

示している。なお、図１２の中間プログラムには、冗長な命令が含まれるが、これらの冗長さは、従来から提案されている最適化処理を行うことで除去できる。従来の各種最適化方法については、例えば文献５（「コンパイラＩＩ　原理・技法・ツール」、Ａ．Ｖ．エイホ他著、原田賢一訳、サイエンス社、１９９０）の第７７２頁～第７９０頁に説明されている。

【０１３７】これらの最適化処理は、レジスタ割り当て前に施すことが好ましいため、図２において、好ましくは、ＦＯＲＫ箇所決定部２１よりも手前、例えば中間プログラム入力部２０の内部に実装される。

【０１３８】［実施の形態２］次に本発明の第２の実施の形態について、図面を参照して詳細に説明する。本発明の第２の実施の形態では、プロファイル情報を参照して、ＦＯＲＫ先及びＦＯＲＫ時のデータ依存対処方法を選択する点に特徴があるが、その他の点は、前記第１の実施の形態と、基本的に同一とされる。

【０１３９】図１を参照すると、本発明の第２の実施の形態においては、まずプログラム変換装置２を用いて原始プログラム１から目的プログラム３を生成する。このとき、プログラム変換装置２は、機械命令アドレスとプログラム変換装置２内で用いる中間プログラムとの対応をとるための情報を、目的プログラム３の中に埋め込む。目的プログラム実行装置４は、この目的プログラム３を実行し、その際に収集した実行情報をプロファイル情報ファイル５として出力する。

【０１４０】次に、再度プログラム変換装置２を用いて、原始プログラム１を並列化し、目的プログラム３を生成する。このとき、プロファイル情報ファイル５を用いて、より実行時性能の高い目的プログラム３を生成する。

【０１４１】プログラム変換装置２の構成は、前記第１の実施の形態とほぼ同様であるが、図２における、ＦＯＲＫ箇所決定部２１と、命令並べ換え部２３の動作が、前記第１の実施の形態とは異なり、プロファイル情報を参照しながら、より実行時性能の期待できる並列化処理を行う。

【０１４２】本発明の第２の実施の形態におけるＦＯＲＫ箇所決定部２１の動作概要は、図３を参照して説明した前記第１の実施の形態と同じである。即ち、ＦＯＲＫ箇所決定部２１は、与えられた中間プログラム中の各関数毎に、図３のステップ２６で当該関数内でレジスタ割り当てを試行した後、ステップ２７で、当該関数内に含まれる各条件分岐命令に対して、ＦＯＲＫ箇所決定処理を行う。

【０１４３】図１３は、各条件分岐命令に対するＦＯＲＫ箇所決定処理（図３のステップ２６）の第２の実施の形態を示す流れ図である。

【０１４４】ステップ５０では、当該条件分岐命令が入力中間プログラム中のループ構造の戻り枝に相当する分岐命令であるか否かを判定する。この処理は、前記第１の実施の形態のステップ３１（図４参照）と同じものである。

【０１４５】もし、ループ構造の戻り枝なら、ステップ５６で、その戻り枝方向をＦＯＲＫ先として仮決定する。これはＦＯＲＫするとすれば、こちらの方向を、ＦＯＲＫ先とするのがよい、という意味であり、実際にＦＯＲＫすることにするか否かは後段で決定する。

【０１４６】ステップ５１では、入力したプロファイル情報に基づいて、当該条件分岐命令のｔａｋｅｎ（分岐成立）側／ｆａｌｌｔｈｒｏｕｇｈ側が選択される確率を計算する。

【０１４７】ステップ５２では、ステップ５１で計算した、２つの確率間に一定以上の違いがあるかどうかを判定する。

【０１４８】分岐確率の違いが、その判断基準を越えていれば、ステップ５７にて、確率の高い側を、ＦＯＲＫ先として仮決定する。

【０１４９】ステップ５３では、条件分岐の２つの分岐先各々に対して、データ依存の距離の最小値を計算する。この処理は、前記第１の実施の形態におけるステップ３２（図４参照）と同じものである。

【０１５０】ステップ５４では、ステップ５３で、条件分岐の両側に関して求めた２つのデータ依存の距離の最小値を比較し、両者に一定以上の差があるか否かを判定する。

【０１５１】一定以上の差があるか、又はデータ依存がない場合、ステップ５５にて、データ依存の距離の最小値の大きい側の分岐方向をＦＯＲＫ先として決定する。これは、前記第１の実施の形態のステップ３３（図４参照）と同様の処理である。

【０１５２】ステップ５８及びステップ５９では、ステップ５６ないしステップ５７で仮決定したＦＯＲＫ先に対して、ステップ５３と同様に、データ依存の距離の最小値を計算し、仮決定したＦＯＲＫ先側のデータ依存の距離の最小値が一定以上あるか否かを判定する。

【０１５３】仮決定したＦＯＲＫ先側のデータ依存の距離の最小値が一定以上あるか、メモリを介したデータ依存がなければ、ステップ５６ないしステップ５７で仮決定したＦＯＲＫ先を、正式なＦＯＲＫ先として確定する（ステップ６０）。

【０１５４】ステップ５４ないしステップ５９で、データ依存の距離の最小値が一定水準に満たないと判断された場合には、ステップ６７にて、当該基本ブロックを、ＦＯＲＫ箇所から除外する。これは例えば、当該条件分岐命令に、ＦＯＲＫ対象外マークを付すなどして、後段の処理で並列化対象から外れるようにする。

【０１５５】ステップ５０からステップ６０で、当該条件分岐命令がＦＯＲＫ箇所として決定され、また分岐命令のいずれかの分岐先がＦＯＲＫ先も決定されたあと、

ステップ６１にて、データ依存発生頻度を計算する。即ち、プロファイル情報に基づき、ＦＯＲＫ元の基本ブロック（現在処理対象としている分岐命令を末尾に含む基本ブロック）中で定義された値がＦＯＲＫ先として決定した基本ブロックにて参照される回数の、このパス（現在の基本ブロックからＦＯＲＫ先と決定した基本ブロックへの制御の流れ）を通過した回数に対する比率を計算する。

【０１５６】このデータ依存発生頻度計算においては、図４のステップ３２と同様に、最終的にレジスタに割り当てられる中間項及び変数によるデータ依存は除外し、メモリ上に配置された中間項及び変数のみを計算対象とする。

【０１５７】ステップ６２にて、そのデータ依存発生頻度が一定水準より高いか否かを判断し、もし高ければステップ６５へ、低ければステップ６３へ進む。

【０１５８】ステップ６３では、中間プログラム中で、ＦＯＲＫ元基本ブロックから、ＦＯＲＫ先基本ブロックへのデータ依存を引き起こし得るメモリ上の中間項／変数の個数を数え挙げ、それが一定水準より多いか否かを判定する。この数え挙げは、プロファイル情報によらず、中間プログラム中の命令列を検査して静的に行う。その結果、データ依存箇所数が一定水準より多ければ、ＤＳＰ方式によるＦＯＲＫ（ステップ６４）、そうでなければＢＬＯＣＫ方式によるＦＯＲＫ（ステップ６６）を用いることとし、その情報を、中間プログラム中のＦＯＲＫ命令に付与する。

【０１５９】ステップ６５では、ステップ６３と同様にデータ依存しているメモリ上の変数の個数を数え挙げる。その数が、一定水準より少なければ、ＢＬＯＣＫ方式によるＦＯＲＫ（ステップ６６）を用いるが、一定水準より多ければ、当該基本ブロックはＦＯＲＫ候補から外す（ステップ６７）。

【０１６０】図１４は、本発明の第２の実施の形態における、命令並べ換え部２３の動作を示す図である。命令並べ換え部２３は、ＦＯＲＫ箇所決定部２１が決定したＦＯＲＫ箇所及びＦＯＲＫ時データ保証方法に基づき、ステップ７０からステップ７６の一連の処理を行う。なお、ステップ７０からステップ７６の各処理はすべて中間プログラムに対して行う。これらのステップの説明に現れる命令は、すべて中間プログラム上の対応する命令を指す。

【０１６１】ステップ７０、ステップ７１、ステップ７２は、前記第１の実施の形態で参照した図５のステップ４０、ステップ４１、ステップ４２とそれぞれ同様の処理である。

【０１６２】ステップ７３では、前段のＦＯＲＫ箇所決定部２１が決定した当該ＦＯＲＫ箇所のＦＯＲＫ時データ保証方式がＢＬＯＣＫ方式かＤＳＰ方式かをチェックし、前者ならステップ７４、後者ならステップ７５に進む。

【０１６３】ステップ７４は、前記第１の実施の形態で参照した図５のステップ４３と同様の処理である。即ち、ＦＯＲＫ前のメモリストア文をＦＯＲＫ後に移動すると共に、必要なブロック設定及びブロック解除命令を挿入する。

【０１６４】移動の際には、データ依存関係を検査し、命令実行順序が入れ替わっても演算結果が不変となるもののみ移動させる、という点も、図５のステップ４３の処理と同様である。

【０１６５】ステップ７５もまた、メモリに対応付けられた中間項への代入文を、ＦＯＲＫ命令後に移動させる、という点で、図５のステップ４３と類似する処理であるが、図５のステップ４３では、アクセス対象のメモリアドレスをオペランドとするブロック設定命令及びブロック解除命令を挿入したのに対し、ステップ７５では、データ依存投機モードでＦＯＲＫを行うように、ステップ７１で作成したＦＯＲＫ命令を修正する点が異なる。

【０１６６】ステップ７６は、前記第１の実施の形態で参照した、図５のステップ４４と同様の処理である。

【０１６７】このように、本発明の第２の実施の形態においては、プロファイル情報を用いて、ＦＯＲＫするか否か、及び、ＦＯＲＫ時のデータ依存保証方法を決定する。このため、条件分岐において制御の流れが片方に偏っている場合には、その方向をＦＯＲＫすることで、制御投機ＦＯＲＫの成功確率が高まる。

【０１６８】また、実際に発生したデータ依存の頻度や、依存箇所の数を考慮した並列化を行うことにより、並列化プログラムの実行時のオーバーヘッドを低減させ、より並列化による性能を引き出しやすくしている。

【０１６９】［実施例２］次に、具体的な実施例を用いて、本発明の第２の実施の形態の動作を説明する。図１５は、本発明の並列化装置によって並列化される中間プログラムの一例を示す図である。図１５における記号等の意味は、図８に示した第１の実施例の中で用いられているものと同じである。また、中間プログラム上で用いる制御並列関連命令も、図７に示した前記第１の実施例のものと同じである。

【０１７０】図１を参照すると、本発明の第２の実施例では、プログラム変換装置２は原始プログラム１を並列化せずに目的プログラム３に変換する。その際、プログラム変換装置２は、各基本ブロックの先頭に当該基本ブロックの識別番号に基づく名前をもつラベルを挿入し、そのラベルに関するシンボル情報も目的プログラム３内に含めて出力する。目的プログラムにシンボル情報を埋め込むことは、コンパイラ分野で広く行われており、目的プログラムからシンボル名及びそのシンボルに対応付けられたアドレスを引き出すことができれば、任意の手法を用いることができる。

35

【０１７１】目的プログラム実行装置４は、目的プログラム３を読み込み、埋め込まれたシンボル情報を元に目的プログラム内の基本ブロックの集合を認識する。これにより、プログラム変換装置２の内部の中間プログラムを構成していた基本ブロック集合と区切り方の等価な基本ブロック集合を、目的プログラム実行装置４も認識することができる。

【０１７２】目的プログラム実行装置４は、読み込んだ目的プログラム３を構成する機械命令をソフトウェアで解釈して実行しながら、目的プログラムの振るまい、具体的には、（１）各条件分岐命令の条件が成立した回数と成立しなかった回数、及び、（２）制御フロー上隣接する基本ブロック間でのメモリデータ依存の回数と当該機械命令アドレス、の情報を収集する。

【０１７３】目的プログラム３の実行が完了した後、目的プログラム実行装置４は収集した前記情報の中の機械命令アドレスを基本ブロックの識別番号に変換し、（１）各基本ブロックから制御フロー上でそれに続く各基本ブロックへ制御が流れた回数、及び、（２）制御フロー上隣接する基本ブロック間でメモリデータ依存を引き起こした回数、の内容を含むプロファイル情報ファイル５を出力する。

【０１７４】図１６は、図１５に示した中間プログラムに対応する目的プログラムを、目的プログラム実行装置４に与えたとき、目的プログラム実行装置４が出力したプロファイル情報ファイル５に含まれるプロファイル情報の一部を示した図である。

【０１７５】図１６（Ａ）は、基本ブロック間の分岐回数であり、例えば基本ブロック（Ｂ１１）から、基本ブロック（Ｂ１２）及び（Ｂ１３）への分岐は、各々２０回、１８０回であったことを示している。

【０１７６】図１６（Ｂ）は、基本ブロック間メモリデータ依存回数を表し、例えば基本ブロック（Ｂ１５）でストアした値を基本ブロック（Ｂ１６）でロードした回数がのべ１２０回であったことを示している。

【０１７７】次に、プロファイル情報ファイル５をプログラム変換装置２に与えて、原始プログラム１を並列化する動作について説明する。

【０１７８】図２を参照すると、並列化装置１１は、一度目に原始プログラムを変換したときと全く同じ中間プログラム６が与えられる。

【０１７９】中間プログラム入力部２０は、中間プログラム６を読み込み、フロー解析を行った上で、ＦＯＲＫ箇所決定部２１に渡す。

【０１８０】他方、プロファイル情報入力部２５は、目的プログラム実行装置４が生成したプロファイル情報ファイル５を読み込む。図１５の中間プログラム例を用いた際にプロファイル情報入力部２５が読み込んだプロファイル情報の中身は、図１６に示したものである。

【０１８１】図２を参照すると、ＦＯＲＫ箇所決定部２１が、図１５に示された中間プログラムを受け取り、ＦＯＲＫ箇所の決定を行う。

【０１８２】図３のステップ２５にて、レジスタ割り当てが試行され、ｔ１～ｔ４３の全中間項、及び、変数Ｊ、Ｐには、物理レジスタが割り当てられ、変数Ｋ、Ｘ、Ｙ、Ｚには、メモリ上の領域が割り当てられたとする。

【０１８３】図３のステップ２６では、条件分岐を含む基本ブロック（Ｂ１１）、（Ｂ１３）、（Ｂ１５）が並列化変換の対象となる。

【０１８４】以下、図１３を参照しながら、基本ブロック（Ｂ１１）、（Ｂ１３）、（Ｂ１５）に対するＦＯＲＫ箇所決定処理部２７の動作を説明する。

【０１８５】基本ブロック（Ｂ１１）は、ステップ５０でループ戻り枝ではないと判定される。

【０１８６】ステップ５１にて、図１６（Ａ）に示したプロファイル情報から、分岐確率を計算し、（Ｂ１２）への分岐が１０％、（Ｂ１３）への分岐が９０％と求められる。

【０１８７】ステップ５２では、比率で２倍以上の偏りがあるかどうかを判断基準にとり、（Ｂ１１）からの分岐確率の偏りは十分であると判定し、ステップ５７により（Ｂ１３）がＦＯＲＫ先として仮決定される。

【０１８８】ステップ５８では、メモリデータ依存を調べるが、（Ｂ１１）にはメモリストアがないため、ステップ６０にて（Ｂ１３）がＦＯＲＫ先に決定される。

【０１８９】次にステップ６１で、データ依存発生頻度を求めるが、（Ｂ１１）に関するメモリデータ依存がないため、結局、ステップ６６にて、ＢＬＯＣＫ方式が候補となる。

【０１９０】基本ブロック（Ｂ１３）は、分岐確率が（Ｂ１４）へ１５％、（Ｂ１５）へ８５％である以外は、上の（Ｂ１１）と同様のステップを辿り、結局、ステップ６０にて、（Ｂ１５）がＦＯＲＫ先と決定され、ステップ６６にて、ＢＬＯＣＫ方式が候補となる。

【０１９１】基本ブロック（Ｂ１５）の場合、分岐確率が（Ｂ１６）へ１５％、（Ｂ１７）へ８５％であり、ステップ５２では、分岐確率の偏りが十分あると判定され、ステップ５７で（Ｂ１７）がＦＯＲＫ先に仮決定される。

【０１９２】（Ｂ１５）から（Ｂ１７）へのデータ依存の距離の最小値は６であり、ステップ５９における判定基準をデータ依存の距離４以上とすると、ステップ６０にて（Ｂ１７）がＦＯＲＫ先として決定する。

【０１９３】ステップ６１にて、図１６（Ａ）及び図１６（Ｂ）に示したプロファイル情報を基にして、（Ｂ１５）から（Ｂ１７）へのデータ依存発生頻度を求めると、４／１７０で約２．４％となる。

【０１９４】ステップ６２での発生頻度の判定基準を３０％とすると、この発生頻度は、低いと判断される。

【０１９５】ステップ６３では、データ依存を起こしうるメモリアクセス箇所を中間プログラム中から数え挙げる。

【０１９６】（Ｂ１５）には、ｍｅｍが、左辺中に２箇所あるほか、メモリ上に割り当てられる変数Ｋが、左辺に１箇所現れる。

【０１９７】これらのメモリストアは、すべて（Ｂ１７）でのメモリロードとアドレスが重なる可能性があり、計３箇所のメモリ依存箇所があることになる。

【０１９８】ステップ６３での依存メモリ箇所数の判定基準を３以上とすると、（Ｂ１５）の場合は依存箇所が多いと判定され、ステップ６４にて、ＦＯＲＫ方式としてＤＳＰ方式が候補となる。

【０１９９】ここで、もし、（Ｂ１５）から（Ｂ１６）及び（Ｂ１７）への分岐確率が各々４０％、６０％であったと仮定すると、ステップ５２では、分岐確率の偏りが小さいと判定される。

【０２００】ステップ５３で、データ依存の距離を求めると、（Ｂ１５）から（Ｂ１６）へのデータ依存の距離の最小値は５、（Ｂ１５）から（Ｂ１７）へのデータ依存の距離の最小値も５となり、ステップ５４では、依存距離の差が小さいと判定される。

【０２０１】そのため、ステップ６７にて、（Ｂ１５）は、ＦＯＲＫ箇所の候補から除外されることになる。

【０２０２】次に、図１４を参照して、第２の実施例における命令並べ換え部２３の動作を具体的に説明する。

【０２０３】命令並べ換え部２３は、基本ブロック（Ｂ１１）、（Ｂ１３）、（Ｂ１５）の各条件分岐に対し、図１４に示された一連の処理を行う。

【０２０４】基本ブロック（Ｂ１１）については、ステップ７１で、（Ｂ１３）へのＳＰＦＯＲＫ命令が作られ、ステップ７２で、条件式計算のための命令群がＳＰＦＯＲＫ後に移される。

【０２０５】（Ｂ１１）からのＦＯＲＫでのデータ依存保証は、ＢＬＯＣＫ方式を候補としているが、ブロックすべきメモリアクセスが存在しないため、ステップ７４でＢＬＯＣＫ命令やＲＥＬＥＡＳＥ命令を挿入することはない。

【０２０６】ステップ７６では、レジスタに乗る予定の中間項ｔ１、ｔ２、ｔ３をオペランドとしたＲＤＣＬ命令を挿入する。

【０２０７】基本ブロック（Ｂ１３）も、ＢＬＯＣＫ方式のデータ依存保証を行うため、前述の基本ブロック（Ｂ１１）と同様の流れで処理される。

【０２０８】基本ブロック（Ｂ１５）の場合、データ依存保証をＤＳＰ方式で行うため、図１４のステップ７５で、メモリストア命令群の移動が行われ、データ依存投機モードを指示する命令が挿入される。

【０２０９】具体的には、ｍｅｍ（ｔ１２）、ｍｅｍ（ｔ１７）、及び、変数Ｋへのストアを構成する文が、

ＳＰＦＯＲＫ命令後に移され、その直後に、データ依存投機モードの終了を指示するＤＳＰＯＵＴが挿入される。

【０２１０】またＳＰＦＯＲＫ命令直前には、データ依存投機モードで子スレッドを生成することを指示するＤＳＰＩＮ命令が挿入される。

【０２１１】図１７は、本発明の第２の実施例における、命令並べ換え部２３の処理を終えた後の中間プログラムを示す図である。

【０２１２】なお、これまで説明した各実施の形態は、他のＦＯＲＫ箇所／ＦＯＲＫ先決定方法と組み合わせて実施することも可能である。例えば文献４（「制御並列アーキテクチャ向け自動並列化コンパイル手法」（酒井他、情報処理学会論文誌、Ｖｏｌ．４０、Ｎｏ．５、Ｍａｙ１９９９））の第２０４９頁～第２０５０頁では、ＦＯＲＫ命令が元の条件分岐命令よりも何命令上流に移動できるかというＦＯＲＫブースト値をＦＯＲＫ箇所選定に利用する方法が開示されている。この方法を導入するには、前記第１の実施の形態及び前記第２の実施の形態における、命令並べ換え部２３の中のステップ４４ないしステップ７６の直前に、ＦＯＲＫブースト値によってＦＯＲＫを行うか否かの判定処理を組み込めばよい。

【０２１３】前記第１及び第２の実施の形態における、プログラム変換装置（コンパイラ）２の並列化装置１１におけるＦＯＲＫ箇所決定部２１、レジスタ割り当て部２２、命令並べ換え部２３、中間プログラム出力部２４、プロファイル情報入力部２５は、コンピュータ上で実行されるプログラムにより、その機能・処理が実現される。この場合、該プログラムを記録した記録媒体（ＣＤ－ＲＯＭ、ＤＶＤ（digital versatile disk）、ＦＤ（フロッピー（登録商標））、ＨＤＤ（ハードディスク）、ＭＴ（磁気テープ）、半導体メモリ）から、該プログラム（実行形式）をコンピュータの主記憶にロードして実行するか、あるいは、サーバ等から通信媒体を介して、コンピュータのＨＤＤ等にダウンロードしてインストールし、該プログラムを実行することで、本発明のプログラム変換装置を実施することができる。

【０２１４】

【発明の効果】以上説明したように、本発明によれば下記記載の効果を奏する。

【０２１５】本発明の第１の効果は、中間プログラムレベルでＦＯＲＫ命令を用いた並列化を的確に行える、ということである。

【０２１６】その理由は、本発明においては、レジスタ割り当て部は並列化よりも後段に位置するにも関わらず、並列化処理にてレジスタ割り当てを試行し、各中間項がレジスタに乗るのかメモリ領域に格納されるのかの予測ができるからである。

【０２１７】本発明の第２の効果は、ＦＯＲＫ命令を用いて並列実行した場合の性能が向上する、ということで

ある。

【０２１８】その理由は２つある。一つは、本発明においては、ＦＯＲＫ箇所決定部が静的に親子スレッド間でのデータ依存の状況を調べ、データ依存による子スレッド実行の一時停止の可能性が低くなるようにＦＯＲＫ先を選定するからである。もう一つは、ＦＯＲＫ箇所決定部が、プロファイル情報に基づいて、動的なデータ依存発生状況を調べ、データ依存による子スレッド実行の一時停止や再実行の可能性が低くなるようにＦＯＲＫ先を選定するからである。

【図面の簡単な説明】

【図１】本発明の実施の形態の全体構成を示す図である。

【図２】本発明の実施の形態におけるプログラム変換装置の内部構成を示す図である。

【図３】本発明の第１の実施の形態におけるＦＯＲＫ箇所決定部の動作を示す図である。

【図４】本発明の第１の実施の形態におけるＦＯＲＫ箇所決定処理部の動作を示す図である。

【図５】本発明の第１の実施の形態における命令並べ換え部の動作を示す図である。

【図６】本発明の第１の実施の形態において命令が並べ換えられる様子を示す図である。

【図７】本発明の第１及び第２の実施例における中間プログラム上の制御並列関連命令一覧を示す図である。

【図８】本発明の第１の実施例における並列化前の中間プログラムを示す図である。

【図９】本発明の第１の実施例において命令並べ換え途中の中間プログラムを示す図である。

【図１０】本発明の第１の実施例において命令並べ換え途中の中間プログラムを示す図である。

【図１１】本発明の第１の実施例において命令並べ換えを終えた中間プログラムを示す図である。

【図１２】本発明の第１の実施例においてレジスタ割り当てを終えた中間プログラムを示す図である。

【図１３】本発明の第２の実施の形態におけるＦＯＲＫ箇所決定処理部の動作を示す図である。

【図１４】本発明の第２の実施の形態における命令並べ換え部の動作を示す図である。

【図１５】本発明の第２の実施例における並列化前の中間プログラムを示す図である。

【図１６】本発明の第２の実施例におけるプロファイル情報を示す図である。

【図１７】本発明の第２の実施例において命令並べ換えを終えた中間プログラムを示す図である。

【図１８】周知のＭＵＳＣＡＴアーキテクチャのＦＯＲＫ命令を説明するための図である。

【図１９】周知のＭＵＳＣＡＴアーキテクチャのＢＬＯＣＫ方式説明のための図である。

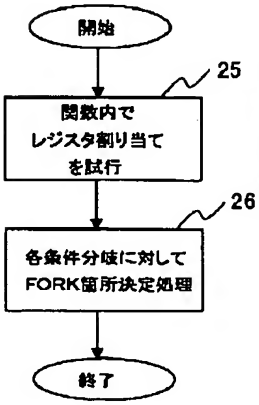
【図２０】周知のＭＵＳＣＡＴアーキテクチャの制御投機モード説明のための図である。

【図２１】周知のＭＵＳＣＡＴアーキテクチャのデータ依存投機モード説明のための図である。

【符号の説明】

１　原始プログラム
２　プログラム変換装置
３　目的プログラム
４　目的プログラム実行装置
５　プロファイル情報ファイル
６　中間プログラム
７　中間プログラム
１０　構文解析装置
１１　並列化装置
１２　コード生成装置
２０　中間プログラム入力部
２１　ＦＯＲＫ箇所決定部
２２　レジスタ割り当て部
２３　命令並べ換え部
２４　中間プログラム出力部
２５　プロファイル情報入力部

【図３】



【図７】

SFFORK i	iから実行開始する投機モード子スレッドを生成
TTERM □ FTERM □ TERMORE	□が真なら自スレッド終了，子スレッド確定 □が偽なら自スレッド終了，子スレッド確定 投機モードの子スレッドを破棄
BLOCK ■ RELEASE ■	■で指定したメモリアドレスをブロック指定 ■で指定したメモリアドレスに設定したブロックを解除
DSPIN DSPOUT	後続のFORKで生成した子スレッドをデータ依存投機モードで生成 子スレッドのデータ依存投機モードを解除
RDCL t,... MDCL t,...	t,...で指定した中間項/変数をレジスタに割り当てるよう指示 t,...で指定した中間項/変数をメモリに割り当てるよう指示

【図１６】

分岐確率

B11	B12: 20	B13: 180
B13	B14: 30	B15: 170
B15	B16: 30	B17: 170

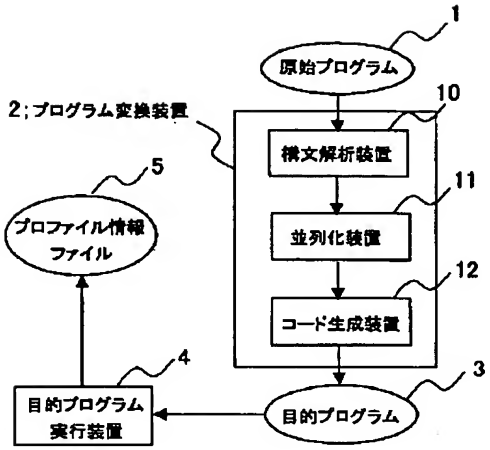
(A)

メモリデータ依存

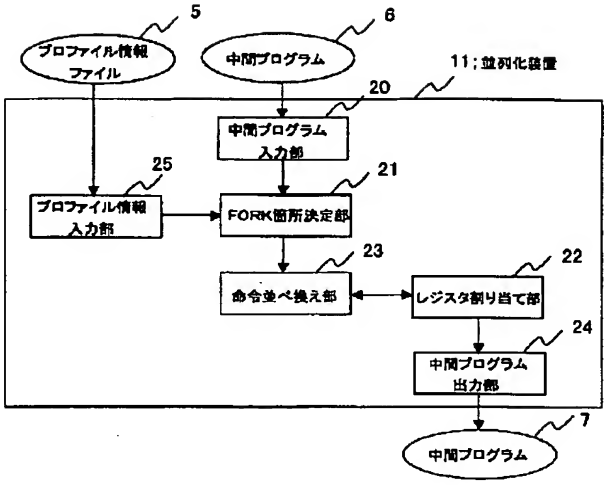
B15 → B16	120
B15 → B17	4

(B)

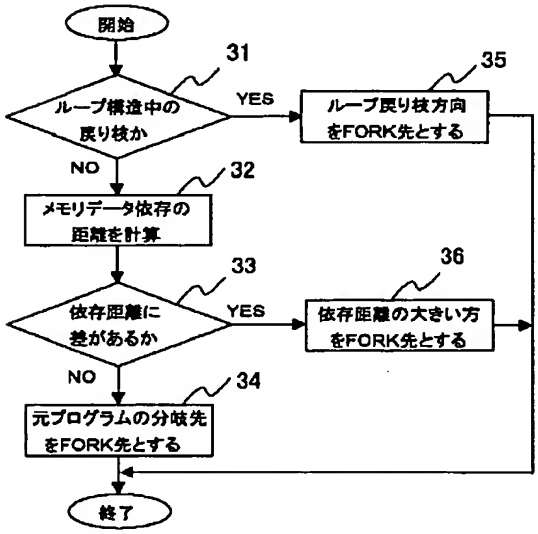
【図１】



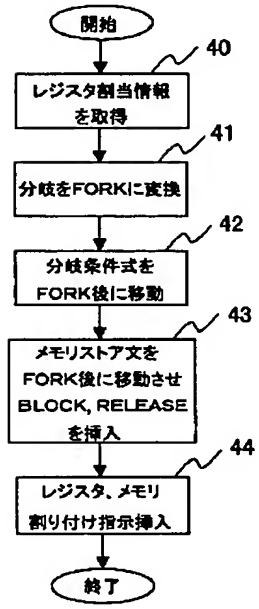
【図２】



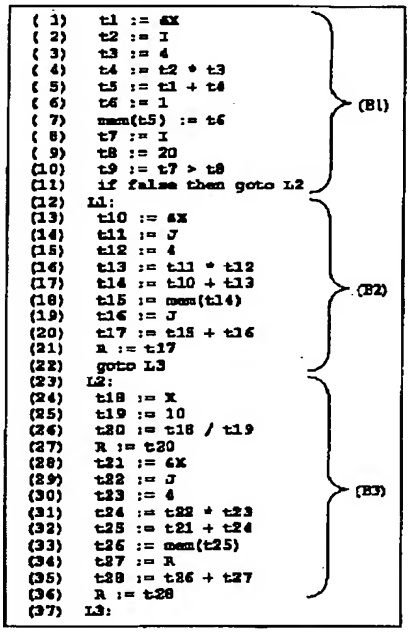
【図４】



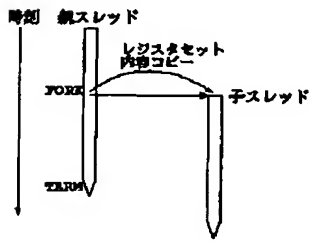
【図５】



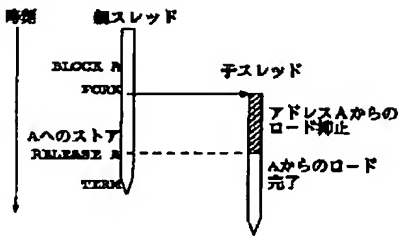
【図８】



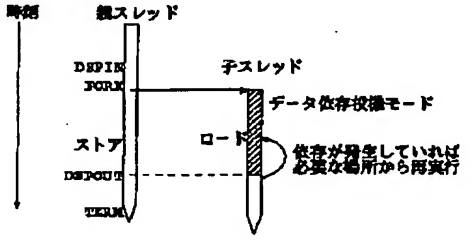
【図１８】



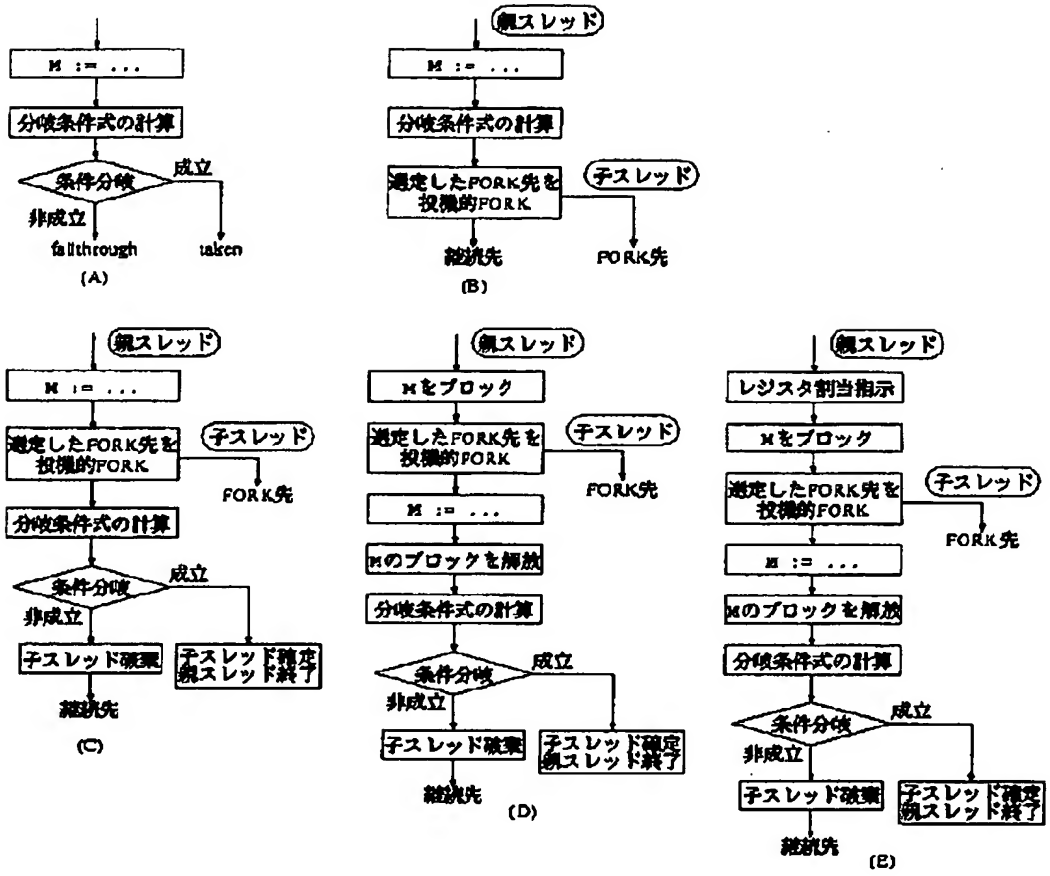
【図１９】



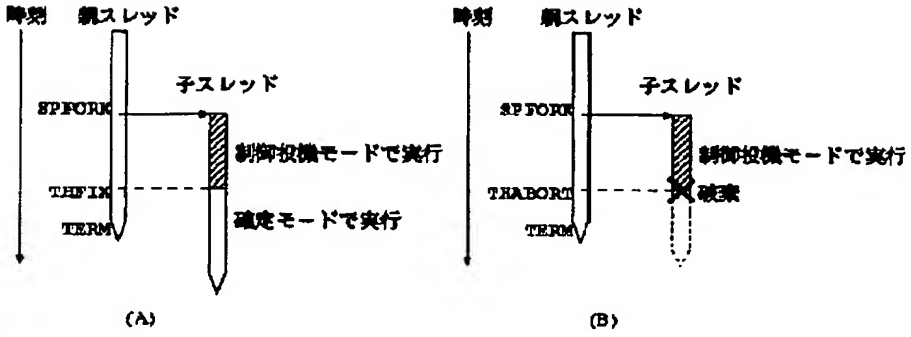
【図２１】



【図６】


M := ...
分岐条件式の計算
条件分岐
成立
非成立
fallthrough
taken
(A)
親スレッド
M := ...
分岐条件式の計算
選定したFORK先を投機的FORK
子スレッド
継続先
FORK先
(B)
親スレッド
M := ...
選定したFORK先を投機的FORK
子スレッド
FORK先
分岐条件式の計算
条件分岐
成立
非成立
子スレッド破棄
子スレッド確定親スレッド終了
継続先
(C)
親スレッド
Mをブロック
選定したFORK先を投機的FORK
子スレッド
FORK先
M := ...
Mのブロックを解放
分岐条件式の計算
条件分岐
成立
非成立
子スレッド破棄
子スレッド確定親スレッド終了
継続先
(D)
親スレッド
レジスタ割当指示
Mをブロック
選定したFORK先を投機的FORK
子スレッド
FORK先
M := ...
Mのブロックを解放
分岐条件式の計算
条件分岐
成立
非成立
子スレッド破棄
子スレッド確定親スレッド終了
継続先
(E)

【図２０】


時刻
親スレッド
子スレッド
SPFORK
制御投機モードで実行
THFIX
TERM
確定モードで実行
(A)
時刻
親スレッド
子スレッド
SPFORK
制御投機モードで実行
THABORT
破棄
TERM
(B)

【図９】

```
(51)   t1 := &X
(52)   t2 := I
(53)   t3 := 4
(54)   t4 := t2 * t3
(55)   t5 := t1 + t4
(56)   t6 := 1
(57)   mem(t5) := t6
(58)   SPFORK L2
(59)   t7 := I
(60)   t8 := 20
(61)   t9 := t7 > t8
(62)   FTERM
(63)   THABORT
(64)   goto L1                    (B1)
(65)  L1:
(66)   t10 := &X
(67)   t11 := J
(68)   t12 := 4
(69)   t13 := t11 * t12
(70)   t14 := t10 + t13
(71)   t15 := mem(t14)            (B2)
(72)   t16 := J
(73)   t17 := t15 + t16
(74)   R := t17
(75)   goto L3
(76)  L2:
(77)   t18 := K
(78)   t19 := 10
(79)   t20 := t18 / t19
(80)   R := t20
(81)   t21 := &X
(82)   t22 := J
(83)   t23 := 4                   (B3)
(84)   t24 := t22 * t23
(85)   t25 := t21 + t24
(86)   t26 := mem(t25)
(87)   t27 := R
(88)   t28 := t26 + t27
(89)   R := t28
(90)  L3:
```

【図１０】

```
(101)   t1 := &X
(102)   t2 := I
(103)   t3 := 4
(104)   t4 := t2 * t3
(105)   t5 := t1 + t4
(106)   BLOCK t5
(107)   SPFORK L2
(108)   t6 := 1                   (B1)
(109)   mem(t5) := t6
(110)   RELEASE t5
(111)   t7 := I
(112)   t8 := 20
(113)   t9 := t7 > t8
(114)   FTERM
(115)   THABORT
(116)   goto L1
(117)  L1:
(118)   t10 := &X
(119)   t11 := J
(120)   t12 := 4
(121)   t13 := t11 * t12
(122)   t14 := t10 + t13          (B2)
(123)   t15 := mem(t14)
(124)   t16 := J
(125)   t17 := t15 + t16
(126)   R := t17
(127)   goto L3
(128)  L2:
(129)   t18 := K
(130)   t19 := 10
(131)   t20 := t18 / t19
(132)   R := t20
(133)   t21 := &X                 (B3)
(134)   t22 := J
(135)   t23 := 4
(136)   t24 := t22 * t23
(137)   t25 := t21 + t24
(138)   t26 := mem(t25)
(139)   t27 := R
(140)   t28 := t26 + t27
(141)   R := t28
(142)  L3:
```

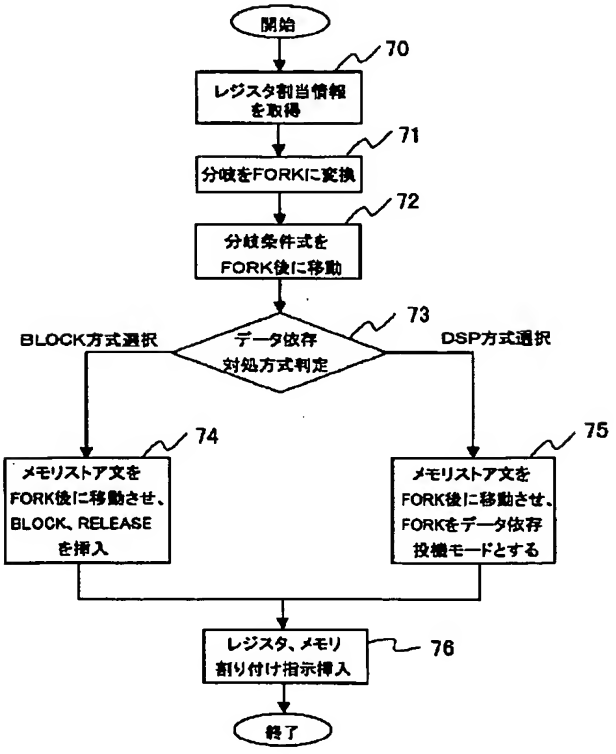
【図１１】

```
(201)   EDCL t1-t9
(202)   EDCL I
(203)   EDCL K
(204)   t1 := &X
(205)   t2 := I
(206)   t3 := 4
(207)   t4 := t2 * t3
(208)   t5 := t1 + t4
(209)   BLOCK t5
(210)   SPFORK L2                 (B1)
(211)   t6 := 1
(212)   mem(t5) := t6
(213)   RELEASE t5
(214)   t7 := I
(215)   t8 := 20
(216)   t9 := t7 > t8
(217)   FTERM
(218)   THABORT
(219)   goto L1
(220)  L1:
(221)   EDCL t10-t17
(222)   EDCL R
(223)   EDCL K, J
(224)   t10 := &X
(225)   t11 := J
(226)   t12 := 4
(227)   t13 := t11 * t12          (B2)
(228)   t14 := t10 + t13
(229)   t15 := mem(t14)
(230)   t16 := J
(231)   t17 := t15 + t16
(232)   R := t17
(233)   goto L3
(234)  L2:
(235)   EDCL t18-t28
(236)   EDCL R
(237)   EDCL K, J
(238)   t18 := K
(239)   t19 := 10
(240)   t20 := t18 / t19
(241)   R := t20
(242)   t21 := &X                 (B3)
(243)   t22 := J
(244)   t23 := 4
(245)   t24 := t22 * t23
(246)   t25 := t21 + t24
(247)   t26 := mem(t25)
(248)   t27 := R
(249)   t28 := t26 + t27
(250)   R := t28
(251)  L3:
```

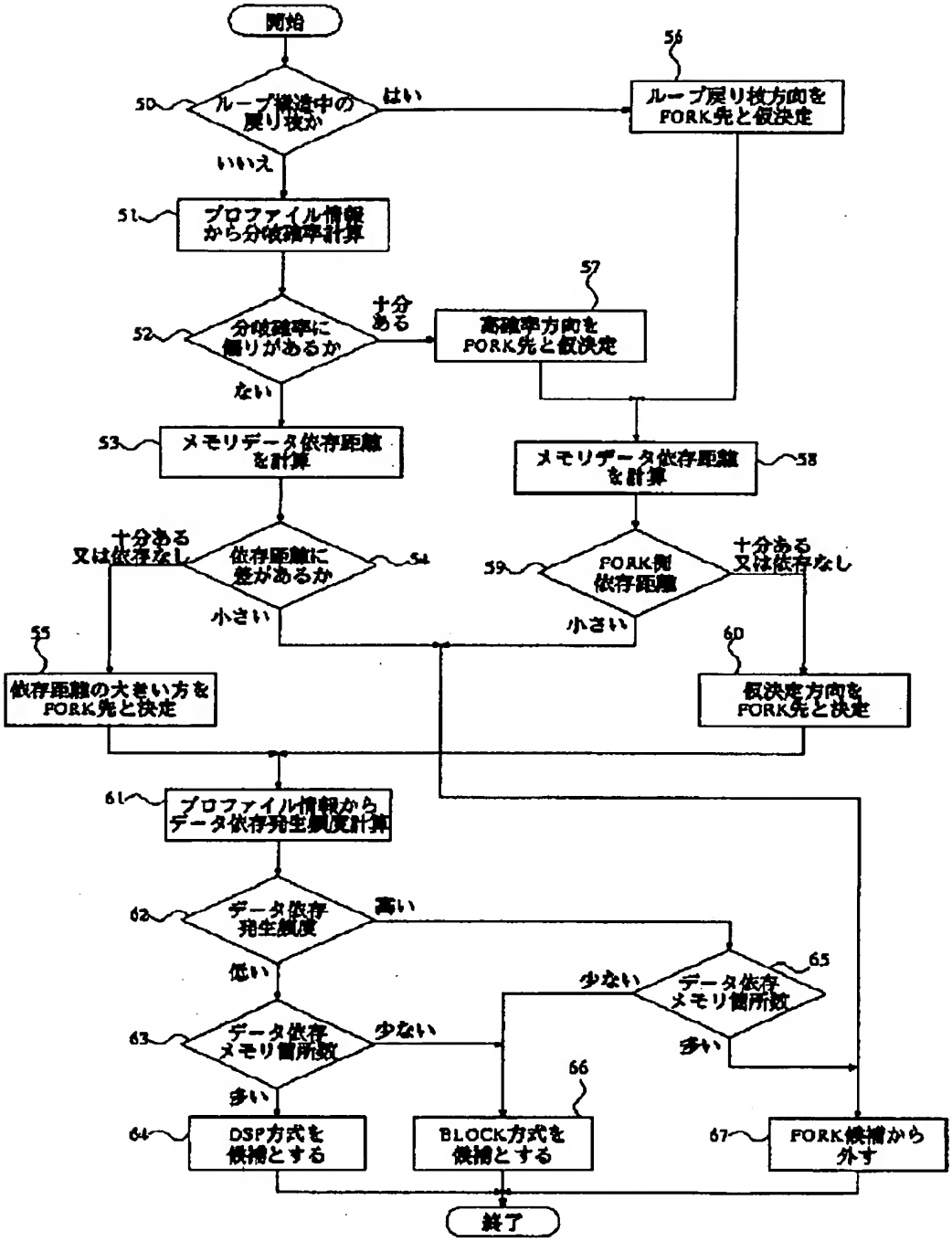
【図１２】

```
(255)   r21 := &X
(256)   r22 := r11
(257)   r23 := 4
(258)   r24 := r22 * r23
(259)   r25 := r21 + r24
(260)   BLOCK r25
(261)   SPFORK L2
(262)   r26 := 1
(263)   mem(r25) := r26
(264)   RELEASE r25
(265)   r27 := r11
(266)   r28 := 20
(267)   r29 := r27 > r28
(268)   FTERM r29
(269)   THABORT
(270)   goto L1
(271)  L1:
(272)   r20 := &X
(273)   r21 := mem(&J)
(274)   r22 := 4
(275)   r23 := r21 * r22
(276)   r24 := r20 + r23
(277)   r25 := mem(r24)
(278)   r26 := mem(&J)
(279)   r27 := r25 + r26
(280)   r12 := r27
(281)   goto L3
(282)  L2:
(283)   r20 := r13
(284)   r21 := 10
(285)   r22 := r20 / r21
(286)   r12 := r22
(287)   r23 := &X
(288)   r24 := mem(&J)
(289)   r25 := 4
(290)   r26 := r24 * r25
(291)   r27 := r23 + r26
(292)   r28 := mem(r27)
(293)   r29 := r12
(294)   r30 := r28 + r29
(295)   r12 := r30
(296)  L3:
```

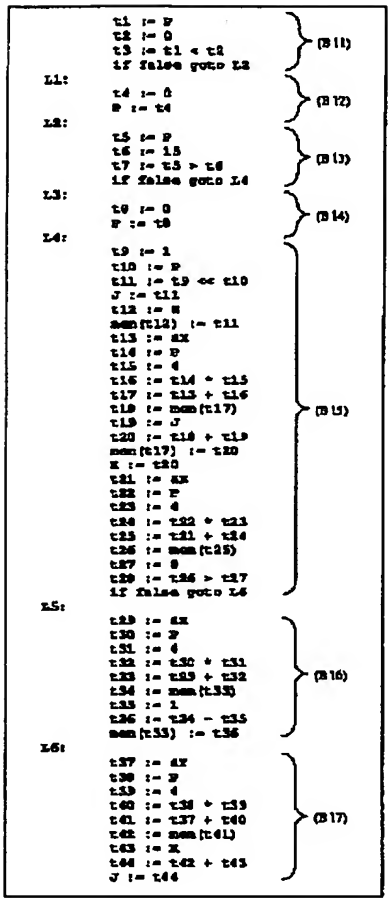
【図１４】



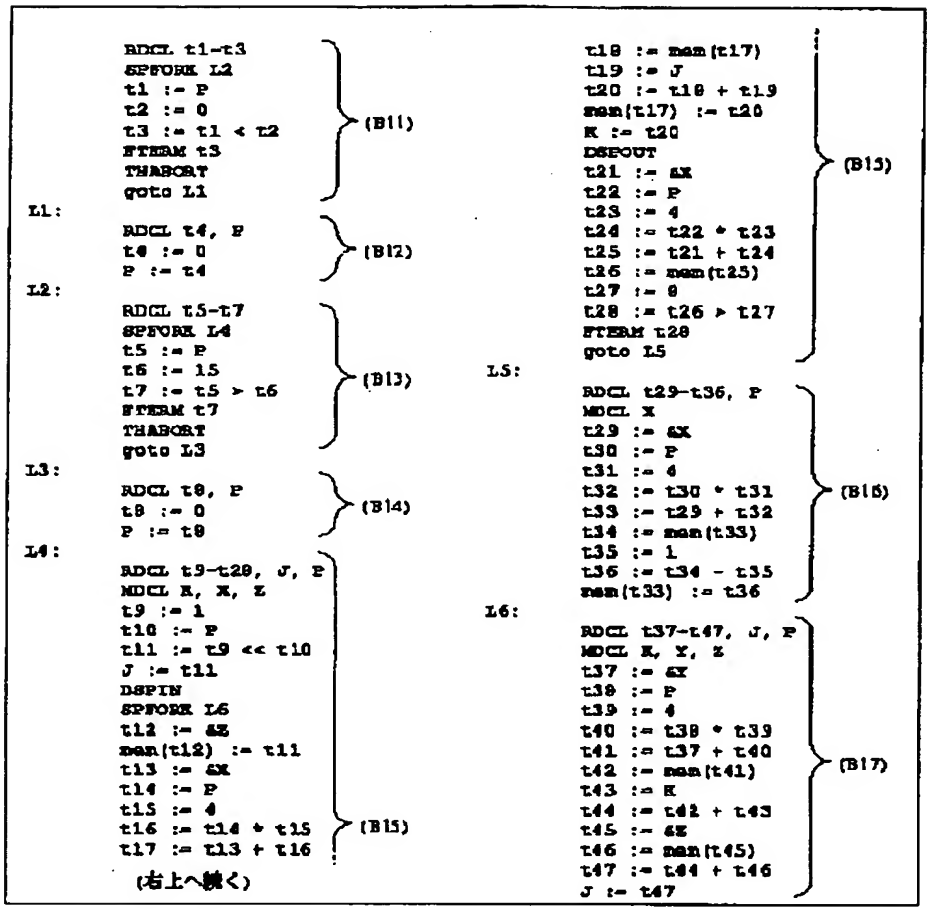
【図１３】


開始
50
ループ構造中の戻り枝か
はい
いいえ
56
ループ戻り枝方向をFORK先と仮決定
51
プロファイル情報から分岐確率計算
52
分岐確率に偏りがあるか
十分ある
ない
57
高確率方向をFORK先と仮決定
53
メモリデータ依存距離を計算
58
メモリデータ依存距離を計算
十分ある又は依存なし
依存距離に差があるか
54
小さい
59
FORK側依存距離
十分ある又は依存なし
小さい
55
依存距離の大きい方をFORK先と決定
60
仮決定方向をFORK先と決定
61
プロファイル情報からデータ依存発生頻度計算
62
データ依存発生頻度
高い
低い
63
データ依存メモリ箇所数
少ない
多い
65
データ依存メモリ箇所数
少ない
多い
66
64
DSP方式を候補とする
BLOCK方式を候補とする
67
FORK候補から外す
終了

【図１５】


t1 := P
t2 := 0
t3 := t1 < t2
if false goto L2
(B11)
L1:
t4 := 8
P := t4
(B12)
L2:
t5 := P
t6 := 15
t7 := t5 > t6
if false goto L4
(B13)
L3:
t8 := 0
P := t8
(B14)
L4:
t9 := 1
t10 := P
t11 := t9 << t10
J := t11
t12 := H
mem(t12) := t11
t13 := ax
t14 := P
t15 := 4
t16 := t14 * t15
t17 := t13 + t16
t18 := mem(t17)
t19 := J
t20 := t18 + t19
mem(t17) := t20
H := t20
t21 := ax
t22 := P
t23 := 4
t24 := t22 * t23
t25 := t21 + t24
t26 := mem(t25)
t27 := 8
t28 := t26 > t27
if false goto L6
(B15)
L5:
t29 := ax
t30 := P
t31 := 4
t32 := t30 * t31
t33 := t29 + t32
t34 := mem(t33)
t35 := 1
t36 := t34 - t35
mem(t33) := t36
(B16)
L6:
t37 := ax
t38 := P
t39 := 4
t40 := t38 * t39
t41 := t37 + t40
t42 := mem(t41)
t43 := H
t44 := t42 + t43
J := t44
(B17)

【図１７】


RDCL t1-t3
SPFORK L2
t1 := P
t2 := 0
t3 := t1 < t2
FTERM t3
THABORT
goto L1
(B11)
L1:
RDCL t4, P
t4 := 0
P := t4
(B12)
L2:
RDCL t5-t7
SPFORK L4
t5 := P
t6 := 15
t7 := t5 > t6
FTERM t7
THABORT
goto L3
(B13)
L3:
RDCL t8, P
t8 := 0
P := t8
(B14)
L4:
RDCL t9-t20, J, P
MDCL K, X, Z
t9 := 1
t10 := P
t11 := t9 << t10
J := t11
DSPIN
SPFORK L6
t12 := &Z
mem(t12) := t11
t13 := &X
t14 := P
t15 := 4
t16 := t14 * t15
t17 := t13 + t16
(B15)
(右上へ続く)
t18 := mem(t17)
t19 := J
t20 := t18 + t19
mem(t17) := t20
K := t20
DSPOUT
t21 := &X
t22 := P
t23 := 4
t24 := t22 * t23
t25 := t21 + t24
t26 := mem(t25)
t27 := 8
t28 := t26 > t27
FTERM t28
goto L5
(B15)
L5:
RDCL t29-t36, P
MDCL X
t29 := &X
t30 := P
t31 := 4
t32 := t30 * t31
t33 := t29 + t32
t34 := mem(t33)
t35 := 1
t36 := t34 - t35
mem(t33) := t36
(B16)
L6:
RDCL t37-t47, J, P
MDCL K, Y, Z
t37 := &Y
t38 := P
t39 := 4
t40 := t38 * t39
t41 := t37 + t40
t42 := mem(t41)
t43 := K
t44 := t42 + t43
t45 := &Z
t46 := mem(t45)
t47 := t44 + t46
J := t47
(B17)